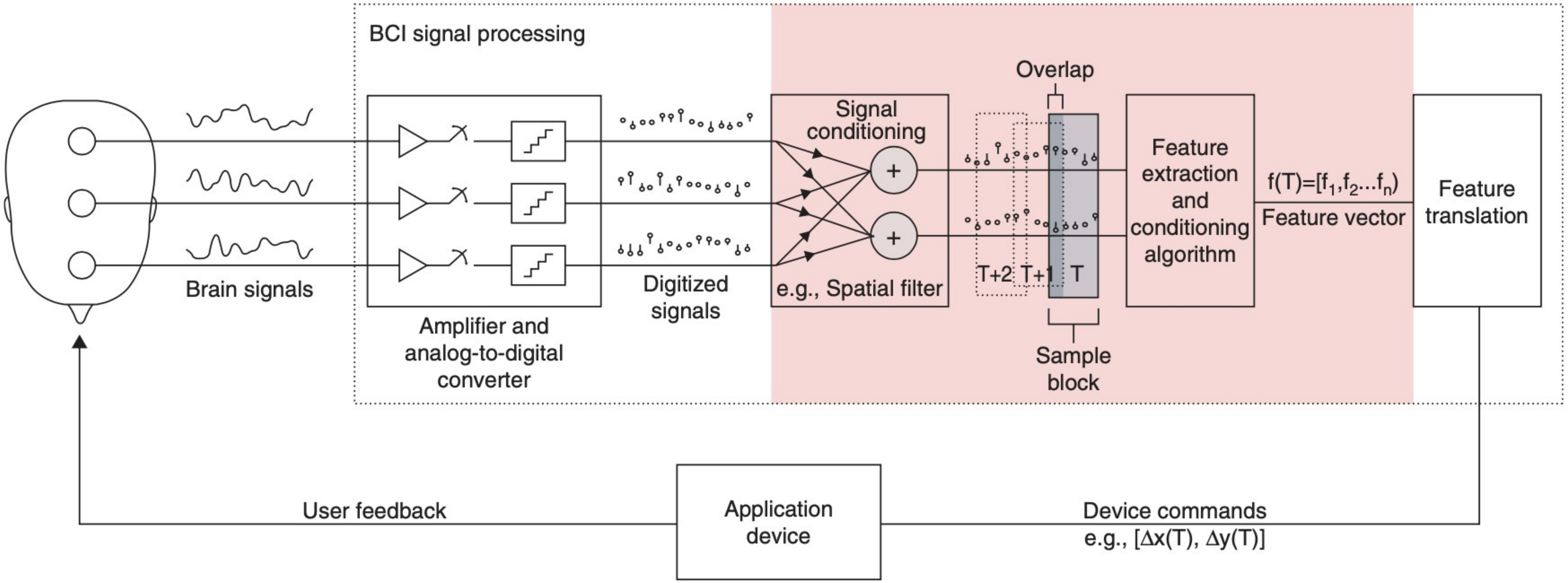
Brain Computer Interaction

Feature Extraction

Features

- The purpose of a BCI is to **detect and quantify characteristics of brain signals** that indicate what the user wants the BCI to do, to translate these measurements in real time into the desired device commands, and to provide concurrent feedback to the user.
- The brain-signal characteristics used for this purpose are called ***signal features*, or simply *features***.
- ***Feature extraction*** is the process of distinguishing the pertinent signal characteristics from extraneous content and representing them in a compact and/or meaningful form, amenable to interpretation by a human or computer.

Overall structure of a BCI



Feature vector

- A **fundamental signal feature** is simply a direct measurement of the signal. They usually provide limited relevant information about typically complex brain signals.
- Thus, it is more common for BCIs to use features that are **linear or nonlinear combinations, ratios, statistical measures, or other transformations of multiple fundamental features** detected at multiple electrodes and/or multiple time points.
- Such complex features, if selected appropriately, can reflect the user's desires more accurately than the fundamental features themselves.
- Most features used in BCI applications are based on **spatial, temporal, and/or spectral analyses** of brain signals or the relationships among them.
- Furthermore, in order to determine the user's wishes as accurately as possible, most BCIs extract a number of features simultaneously. This set of features is referred to as a ***feature vector***.

Feature vector

To be effective for BCI applications, a feature should have the following attributes:

- its spatial, temporal, spectral characteristics, and dynamics can be precisely characterized for an individual user or population of users
- it can be modulated by the user and used in combination with other features to reliably convey the user’s intent
- its correlation with the user’s intent is stable over time and/or can be tracked in a consistent and reliable manner

BCI SIGNAL PROCESSING- Fourier Analysis

- Much of signal-processing theory is rooted in Fourier analysis, which transforms a time-domain (i.e., time on the x-axis) signal into its equivalent frequency-domain (i.e., frequency on the -axis) representation.
- The primary utility of Fourier analysis is to decompose a signal into individual sinusoidal components that can be isolated and evaluated independently.
- Using Fourier analysis, practically any signal can be accurately represented as the sum of a number (possibly an infinite number) of amplitude-scaled and time-shifted sinusoids at specific frequencies.
- In order to model these signals, it is necessary to properly adjust the phase and the magnitude of each sinusoid.

Fourier Analysis

- For an arbitrary signal x(t), the magnitude (scale) and phase (shift) of the sinusoid at each frequency [ω(radians) = 2πf (Hz)] required to represent an arbitrary signal can be determined from the Fourier transform:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt = \int_{-\infty}^{\infty} x(t)[\cos \omega t + j \sin \omega t] dt$$

$$= \underbrace{\int_{-\infty}^{\infty} x(t) \cos \omega t dt}_{a(\omega)} + j \underbrace{\int_{-\infty}^{\infty} x(t) \sin \omega t dt}_{b(\omega)}$$

$$= a(\omega) + jb(\omega)$$

The magnitude and phase for each sinusoidal component are given as:

$$Magnitude: \quad |X(\omega)| = \sqrt{a^2(\omega) + b^2(\omega)}$$

$$Phase: \quad \theta = \arg(X(\omega)) = \tan^{-1}\left(\frac{b(\omega)}{a(\omega)}\right)$$

BCI SIGNAL PROCESSING – Digital Filtering

- Digital filters are central to digital signal processing. They modify the frequency content of a digital signal by attenuating some frequencies (or frequency ranges) and amplifying others. Each successive sample of a digitized signal is passed through a digital filter to produce a new value as the output.

  - Low pass filter (higher frequencies are attenuated and lower frequencies are preserved)
  - High pass filter (very specific ranges of signal frequencies can be amplified, attenuated, preserved, and/or eliminated)
  - Bandpass filter (A band-pass filter preserves signal power within a specified continuous frequency range, while attenuating signal power outside of this range)
  - Notch filter (A notch filter is the converse of a bandpass filter; it attenuates signal power within a specified continuous frequency range, while preserving signal power outside of this range)

THE THREE STEPS OF FEATURE EXTRACTION

The process of feature extraction is discussed here as a three-step procedure:

- signal conditioning to reduce noise and to enhance relevant aspects of the signals
- extraction of the features from the conditioned signals
- feature conditioning to properly prepare the feature vector for the feature-translation stage

FIRST STEP: SIGNAL CONDITIONING

- The first step of feature extraction is called signal conditioning or preprocessing.
- This step enhances the signal by preemptively eliminating known interference (i.e., artifacts) or irrelevant information, and/or by enhancing spatial, spectral, or temporal characteristics of the signal that are particularly relevant to the application.
- It is common to have some prior knowledge about the general signal characteristics relevant for a particular application, and this knowledge is used in conditioning.

FIRST STEP: SIGNAL CONDITIONING

Signal conditioning can include a number of different procedures that can primarily be categorized as:

- frequency-range prefiltering
- data decimation and normalization
- spatial filtering
  - Data independent spatial filtering (*common- average reference* and *surface Laplacian spatial filters)*
  - Data dependent spatial filtering (PCA, ICA and CSP)
- removal of environmental interference and biological artifacts

SECOND STEP: EXTRACTING THE FEATURES

- ***BLOCK PROCESSING***
  - For most BCI applications, it is highly desirable for the processing to occur in real time. Prior to feature extraction, the incoming signal samples are commonly segmented into consecutive, possibly overlapping, sample blocks.
  - A feature vector is created from the signal samples within each individual sample block. The feature vectors from the successive sample blocks are then fed to the translation algorithm, which produces a device command or user feedback corresponding to each sample block or corresponding to sets of consecutive sample blocks.
  - For efficient online implementation, the length and overlap of these sample blocks should fit the relevant temporal dynamics of the signal, the feature-extraction method, the nature of the application, and the concurrent user feedback, as well as the available processing power.
    - E.g., BCI cursor control
    - P300 response

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***

1. Peak-Picking and Integration
   - Peak-picking simply determines the minimum or maximum value of the signal samples in a specific time block (usually defined relative to a specific preceding stimulus) and uses that value (and possibly its time of occurrence) as the feature(s) for that time block.
   - The signal can be averaged or integrated over all or part of the time block to yield the feature(s) for the block. Some form of averaging or integration is typically preferable to simple peak-picking, especially when the responses to the stimulus are known to vary in latency and/or when unrelated higher-frequency activity is superimposed on the relevant feature

SECOND STEP: EXTRACTING THE FEATURES

- ***TIME (TEMPORAL) FEATURES***

2. Correlation and Template-Matching
    - The similarity of a response to a predefined template might also be used as a feature.

SECOND STEP: EXTRACTING THE FEATURES

Statistical features

Sl. No	Features	Short description
1	MEAN	Mean value
2	STD	Standard deviation
3	MAX VALUE	Maximum positive amplitudes
4	MIN VALUE	Maximum negative amplitudes
5	SKEWNESS	a measure of asymmetry of the distribution
6	KURTOSIS	a measure of flatness of the distribution
7	MEDIAN	the middle value of a set of ordered data

SECOND STEP: EXTRACTING THE FEATURES

Interval or period analysis features.

Sl. No	Features	Short description
8	LINE LENGTH	Line length
9	MEAN VV AMPL	Mean of vertex-to-vertex amplitudes
10	VAR VV AMPL	Variance of vertex-to-vertex amplitudes
11	MEAN VV TIME	Mean of vertex-to-vertex times
12	VAR VV TIME	Variance of vertex-to-vertex times
13	MEAN VV SLOPE	Mean of vertex-to-vertex slope
14	VAR VV SLOPE	Variance of vertex-to-vertex slope
15	ZERO CROSSING	Number of zero crossings in a signal
16	MIN MAX NUMBER	Number of local minima and maxima
17	COEFF OF VARIATION	a statistical measure of the deviation of a variable from its mean, standard deviation divided by mean
18	AMPL RANGE	The difference between the maximum positive and maximum negative Amplitude values

SECOND STEP: EXTRACTING THE FEATURES

Features derived from the first and second derivative.

Sl. No	Features	Short description
19	1st DIFF MEAN	Mean value of the first derivative of the signal
20	1st DIFF MAX	Maximum value of the first derivative of the signal
21	2nd DIFF MEAN	Mean value of the second derivative of the signal
22	2nd DIFF MAX	Maximum value of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

The Hjorth parameters

Sl. No	Features	Short description
23	HJORTH 1	Ability
24	HJORTH 2	Mobility (*σx ′/σx*)
25	HJORTH 3	Complexity

NOTE:

σx ′ is the standard deviation of the first derivative of the signal
σx″ is the standard deviation of the second derivative of the signal

SECOND STEP: EXTRACTING THE FEATURES

- ***FREQUENCY (SPECTRAL) FEATURES***
- Much brain activity manifests itself as continuous amplitude- and frequency-modulated oscillations. Therefore, it is often advantageous to accurately track these changes in the frequency domain. Although the Fourier transform is the most common method for converting from the time domain to the frequency domain, there are several alternatives that have characteristics that are particularly desirable given specific constraints or specific objectives. These include:
    - band power
    - fast Fourier transform (FFT)
    - autoregressive (AR) modeling

SECOND STEP: EXTRACTING THE FEATURES

FFT-based features calculated from the EEG spectra.

Sl. No	Features	Short description
26	FFT DELTA	0.1 - 3 Hz
27	FFT THETA	3 - 7 Hz
28	FFT ALPHA	7 - 12 Hz
29	FFT BETA	12 - 30 Hz
30	FFT GAMMA	30 - 40 Hz
31	FFT WHOLE	0.1 - 40 Hz

SECOND STEP: EXTRACTING THE FEATURES

FFT-based Spectral Features.

Sl. No	Features	Short description
32	FFT DT RATIO	*DELTA / THETA*
33	FFT DA RATIO	*DELTA / ALPHA*
34	FFT TA RATIO	*THETA / ALPHA*
35	FFT DTA RATIO	*(DELTA + THETA) / ALPHA*
36	FFT SEF	Spectral edge frequency (95 % of the total spectral power resides)
37	FFT SP-ROLL OFF	The frequency below which 85 % of the total spectral power resides

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
38	MIN WAV VALUE	Minimum value
39	MAX WAV VALUE	Maximum value
40	MEAN WAV VALUE	Mean value
41	MEDIAN WAV VALUE	Median value
42	STD WAV VALUE	Standard deviation
43	SKEWNESS WAV VALUE	Skewness
44	KURTOSIS WAV VALUE	Kurtosis
45	WAV BAND	Relative energy

SECOND STEP: EXTRACTING THE FEATURES

Wavelet based Features.

Sl. No	Features	Short description
46	ENTROPY SPECTRAL WAV	The spectral entropy
47	1*st* DIFF WAV MEAN	Mean value of the 1*st* derivative
48	1*st* DIFF WAV MAX	Maximum value of the 1*st* derivative
49	2nd DIFF WAV MEAN	Mean value of the 2*nd* derivative
50	2nd DIFF WAV MAX	Maximum value of the 2*nd* derivative
51	ENERGY PERCENT WAV	Percentage of the total energy of a detail/approximation
52	WAV ZERO CROSSING	Zero crossing
53	WAV COEFF OF VARIATION	Coefficient of variation
54	WAV TOTAL ENERGY	Total Energy

SECOND STEP: EXTRACTING THE FEATURES

Other Features.

Sl. No	Features	Short description
55	ENTROPY SPECTRAL	The spectral entropy
56	ENTROPY SHANNON	The Shannon entropy
57	MAX ABS XCORR EEG-EEG	Maximum positive amplitude of auto-correlation or cross-Correlation function
58	MEAN ABS XCORR EEG-EEG	Mean value of auto-correlation or cross-correlation function

THIRD STEP: FEATURE CONDITIONING

- The distributions and the relationships among the features can have a significant effect on the performance of the translation algorithm that follows feature extraction. These effects depend on the characteristics of the particular translation algorithm.

  - ***NORMALIZATION***
  - ***LOG-NORMAL TRANSFORMS***
  - ***FEATURE SMOOTHING***
  - ***PCA AND ICA***

Thank You!

Brain Computer Interaction

Feature Translation

<u>Course Instructors</u>

Dr. Annushree Bablani

Acknowledgements: Dr. Sreeja S R

THIRD STEP: FEATURE CONDITIONING

- The distributions and the relationships among the features can have a significant effect on the performance of the translation algorithm that follows feature extraction. These effects depend on the characteristics of the particular translation algorithm.

  - ***NORMALIZATION***
  - ***LOG-NORMAL TRANSFORMS***
  - ***FEATURE SMOOTHING***
  - ***PCA AND ICA***
  - ***Removing irrelevant and redundant features (MRMR-Maximum Relevant Minimum Redundant)***

Feature Translation

- Discriminant functions
- Regression functions

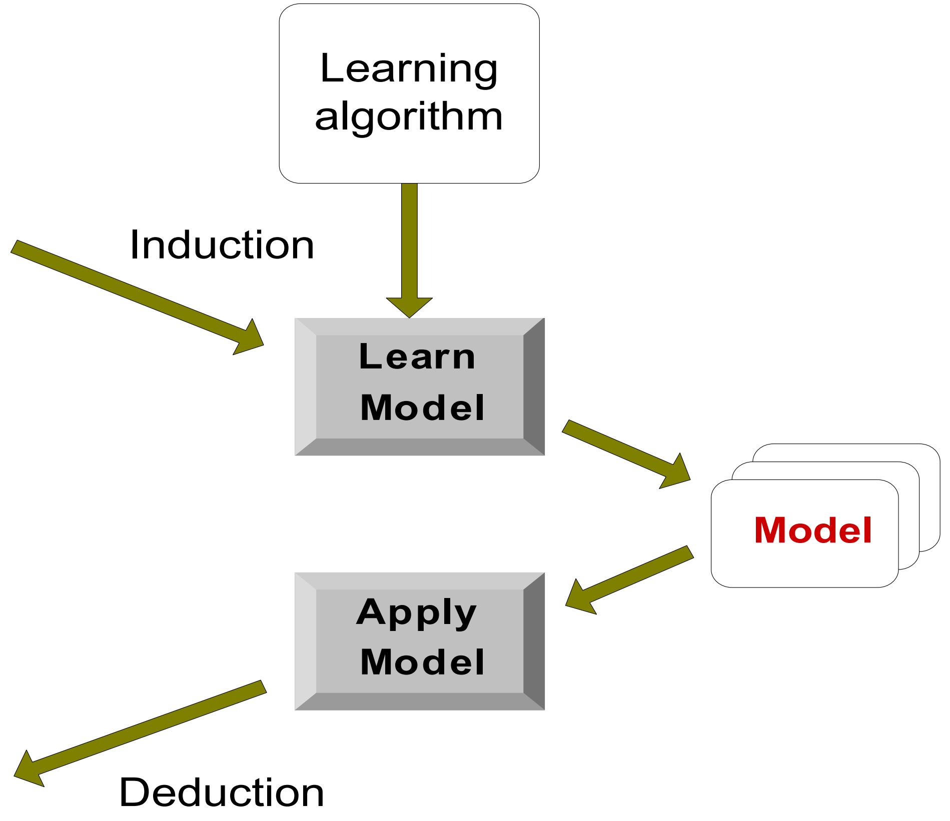
Illustrating Classification Tasks

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

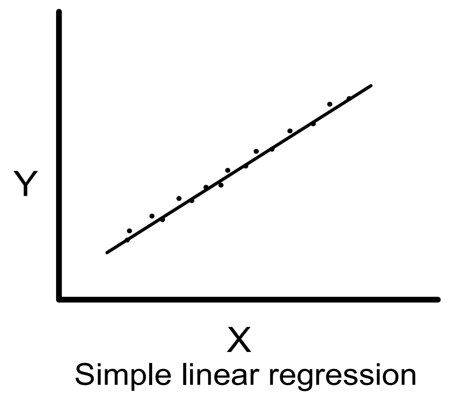
Test Set



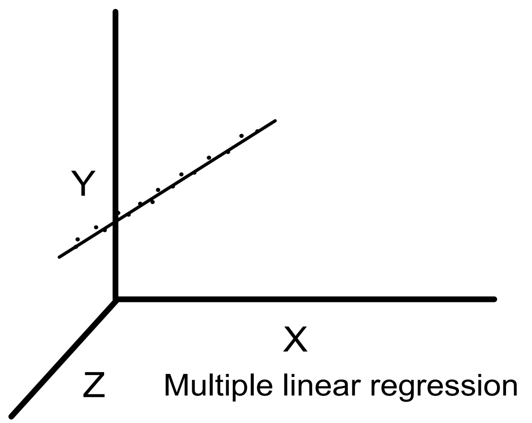
Regression Analysis

- The regression analysis is a statistical method to deal with the formulation of mathematical model depicting relationship amongst variables, which can be used for the purpose of prediction of the values of dependent variable, given the values of independent variables.

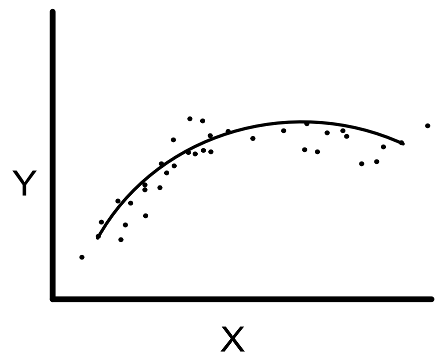
- **Classification of Regression Analysis Models**
  - Linear regression models
    1. Simple linear regression
    2. Multiple linear regression
  - Non-linear regression models



Simple linear regression



Multiple linear regression

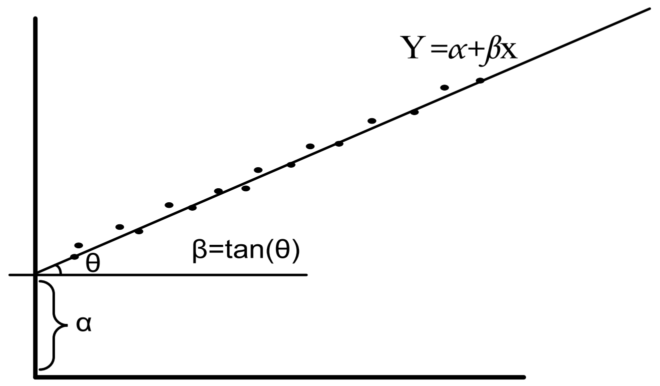


Non-linear regression

Simple Linear Regression Model

In simple linear regression, we have only two variables:

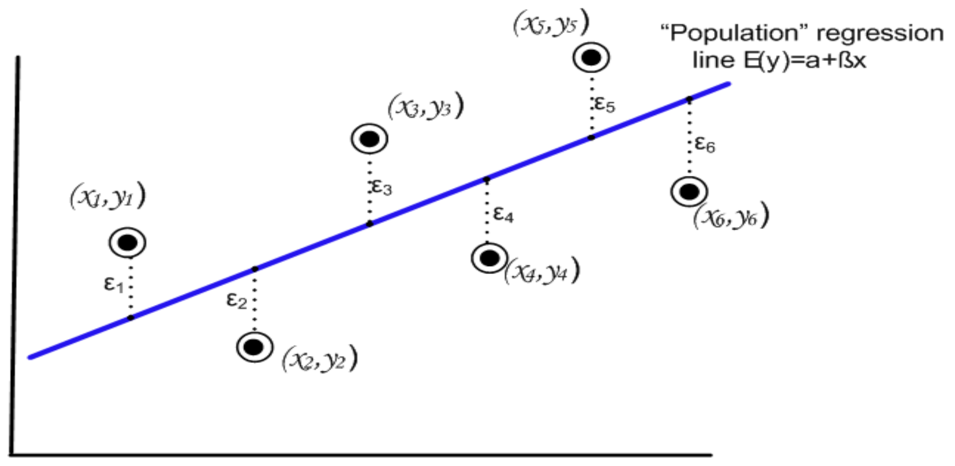
- Dependent variable (also called Response), usually denoted as .
- Independent variable (alternatively called Regressor), usually denoted as .
- A reasonable form of a relationship between the Response and the Regressor is the linear relationship, that is in the form



Note:

- There are infinite number of lines (and hence)
- The concept of regression analysis deal with finding the best relationship between and (and hence best fitted values of) quantifying the strength of that relationship.

Regression Analysis



Given the set of data involving pairs of values, our objective is to find "true" or population regression line such that

Here, is a random variable with and . The quantity is often called the **error variance**.

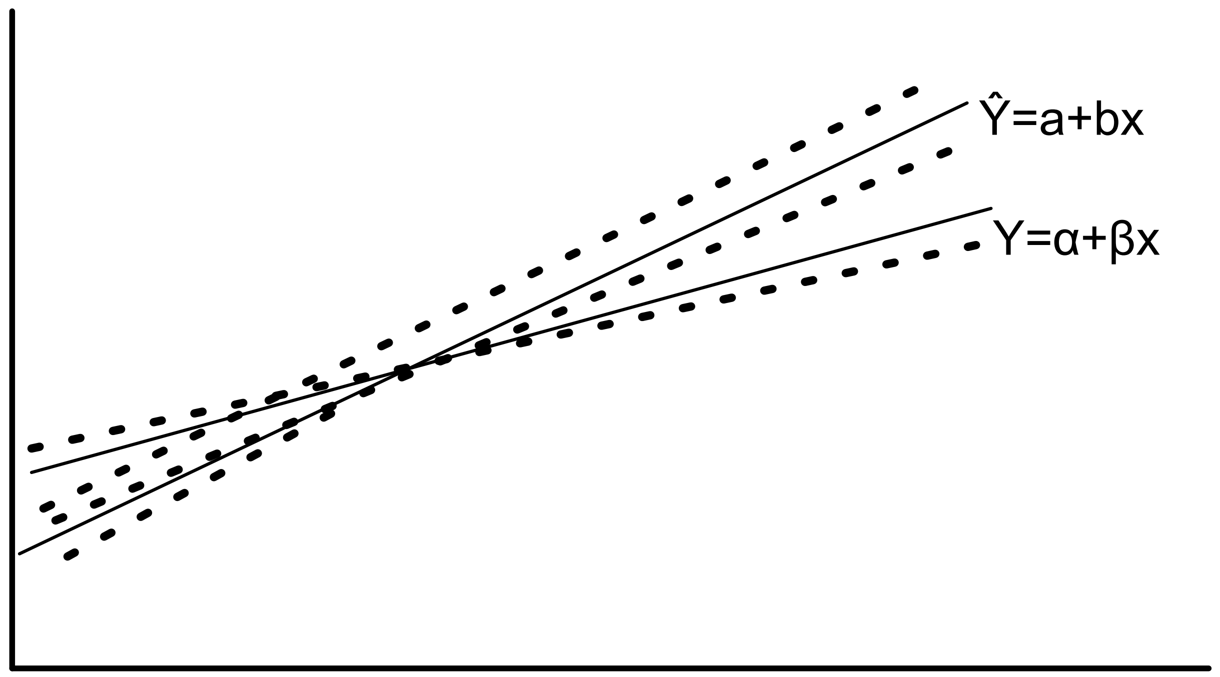
Note:

- implies that at a specific , the values are distributed around the "true" regression line (i.e., the positive and negative errors around the true line is reasonable).
- are called **regression coefficients**.
- values are to be estimated from the data.

True versus Fitted Regression Line

- The task in regression analysis is to estimate the regression coefficients .
- Suppose, we denote the estimates *a* for and *b* for . Then the fitted regression line is

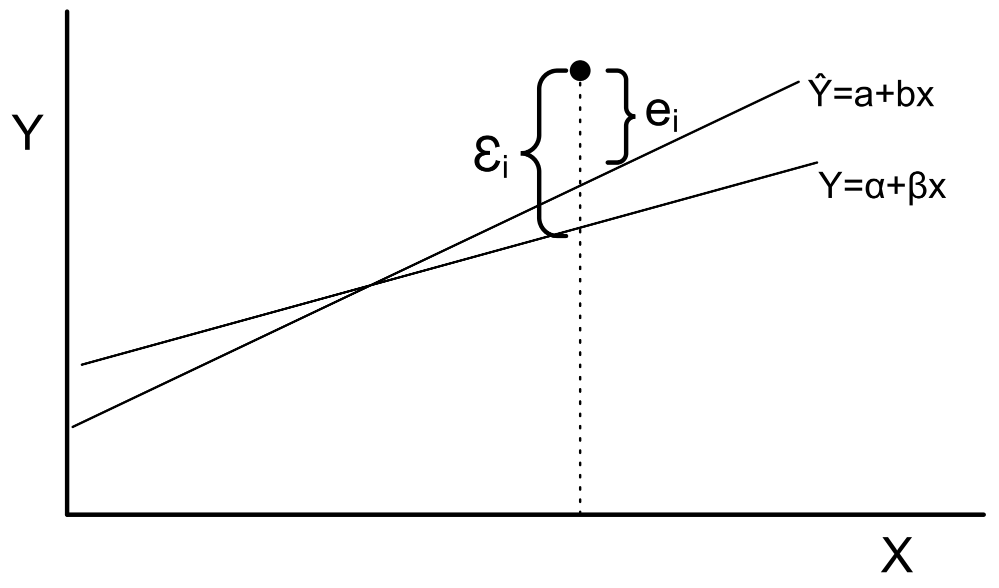
where is the predicted or fitted value.



Least Square Method to estimate

This method uses the concept of residual. A residual is essentially an error in the fit of the model . Thus, residual is

,



Least Square method

- The residual sum of squares is often called **the sum of squares of the errors** about the fitted line and is denoted as SSE

$$\text{SSE} = \quad =$$

- We are to minimize the value of SSE and hence to determine the parameters of *a* and *b*.

- Differentiating SSE with respect to *a* and *b*, we have

For minimum value of SSE, $\quad 0$

0

Least Square method to estimate

Thus we set

+b=

These two equations can be solved to determine the values of and *b*, and it can be calculated that

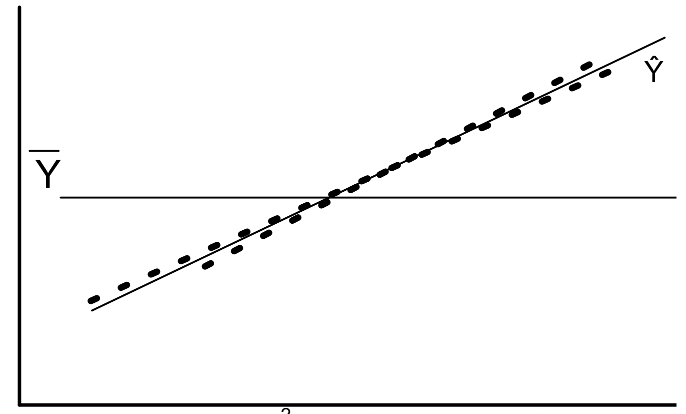
: Measure of Quality of Fit

- A quantity , is called **coefficient of determination** is used to measure the proportion of variability of the fitted model.
- We have
- It signifies the **variability due to error**.
- Now, let us define the total corrected sum of squares, defined as
- SST represents the variation in the response values. The is

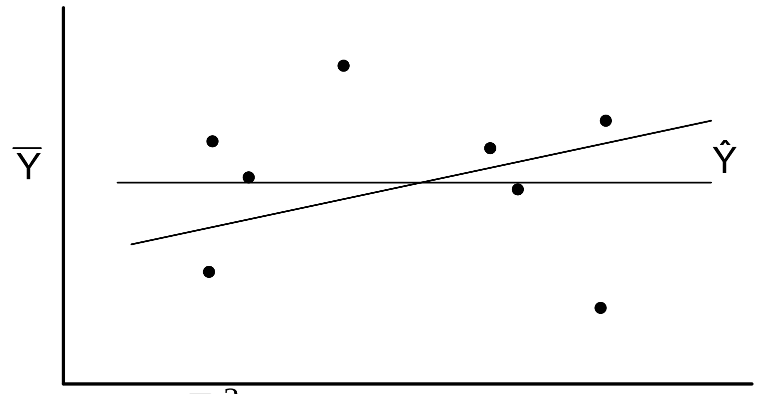
Note:

- If fit is perfect, all residuals are zero and thus = 1.0 (very good fit)
- If SSE is only slightly smaller than SST, then (very poor fit)

: Measure of Quality of Fit



$R^2 \approx 1.0$ (Very good fit)

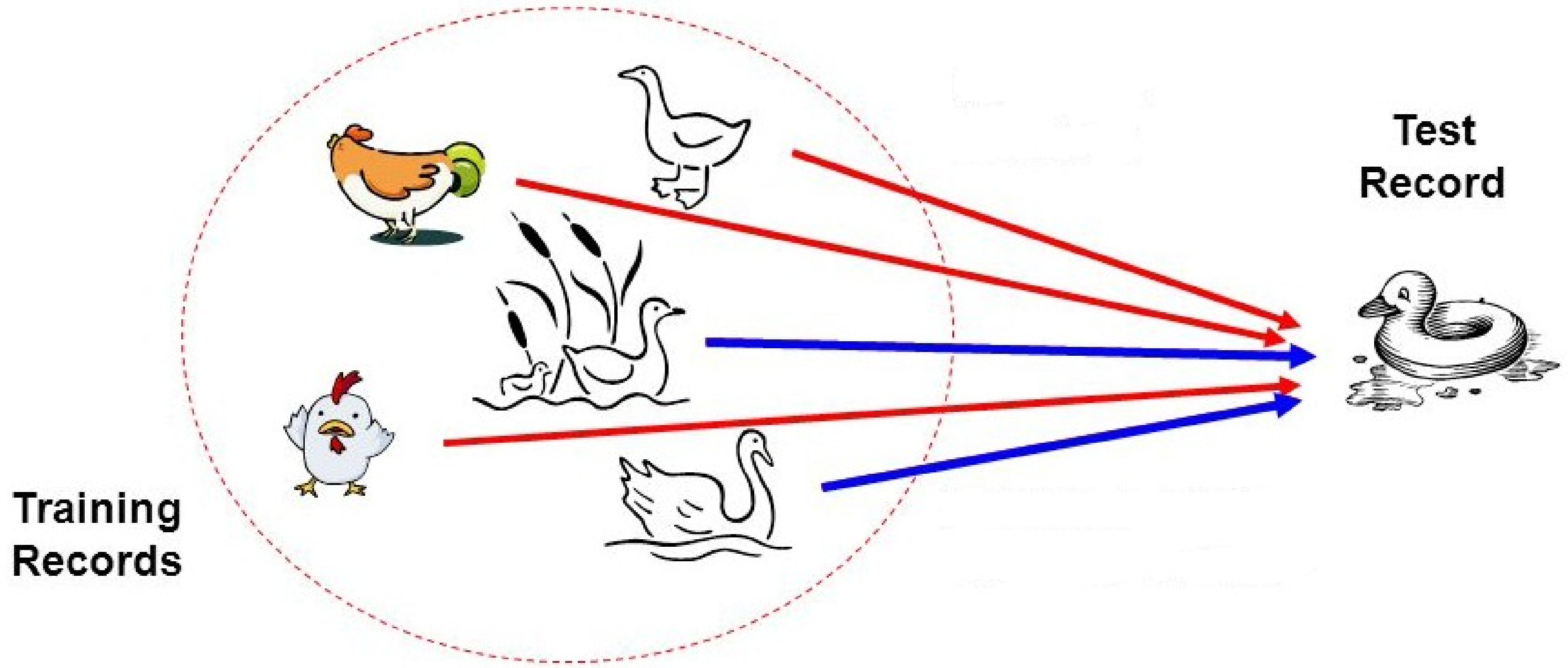


$R^2 \approx 0$ (Very poor fit)

Bayesian Classifier

Bayesian Classifier

- Principle
  - If it walks like a duck, quacks like a duck, then it is probably a duck

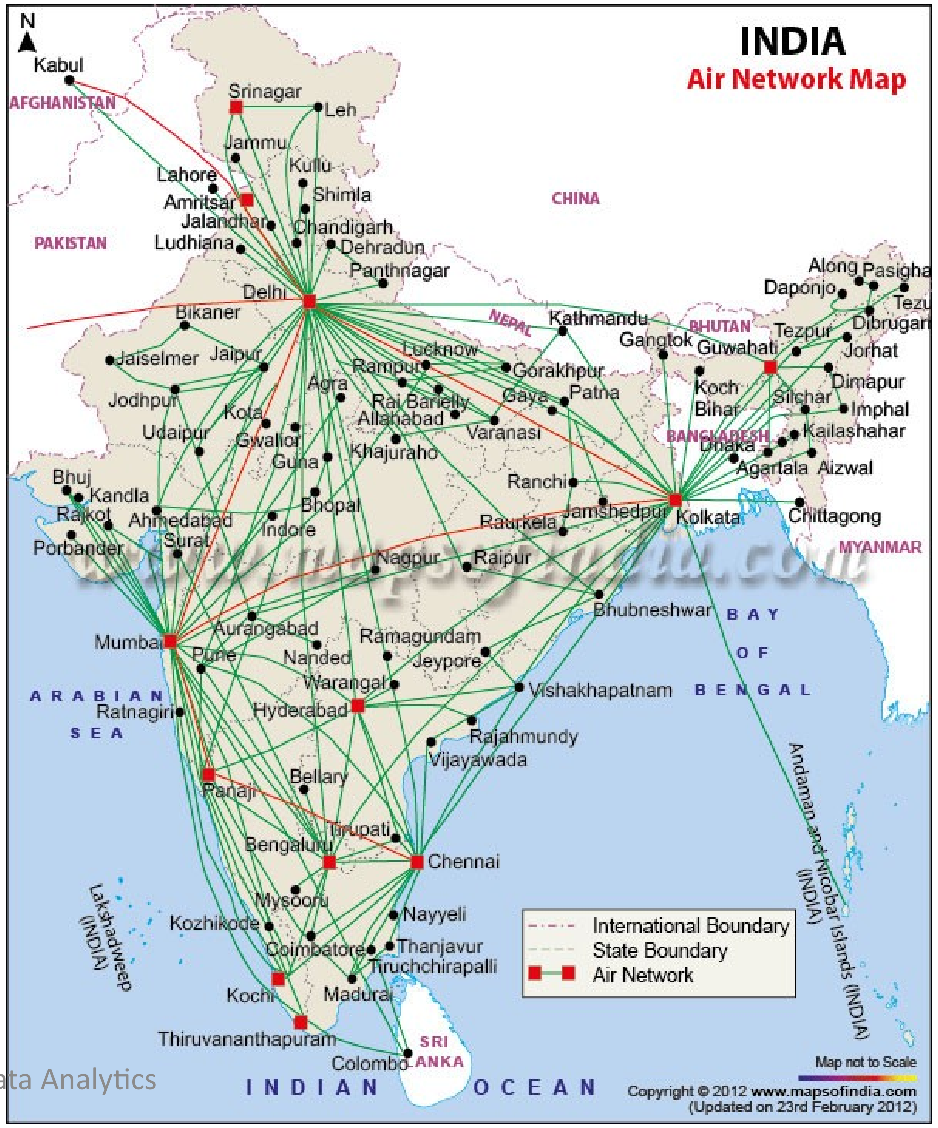


Bayesian Classifier

- A statistical classifier
  - Performs *probabilistic prediction*, *i.e.,* predicts class membership probabilities
- Foundation
  - Based on Bayes' Theorem.
- Assumptions
  1. The classes are mutually exclusive and exhaustive.
  2. The attributes are independent given the class.
- Called "Naïve" classifier because of these assumptions.
  - Empirically proven to be useful.
  - Scales very well.

Example: Bayesian Classification

- **Example 8.2:** Air Traffic Data
  - Let us consider a set observation recorded in a database
    - Regarding the arrival of airplanes in the routes from any airport to New Delhi under certain conditions.



Air-Traffic Data

Days	Season	Fog	Rain	Class

Cond. to next slide…

Air-Traffic Data

Cond. from previous slide…

Days	Season	Fog	Rain	Class

Air-Traffic Data

- In this database, there are four attributes

  A = [Day, Season, Fog, Rain]

  with 20 tuples.
- The categories of classes are:

  C= [On Time, Late, Very Late, Cancelled]

- Given this is the knowledge of data and classes, we are to find most likely classification for any other unseen instance, for example:

Week Day	**Winter**	**High**	**None**	**???**

- Classification technique eventually to map this tuple into an accurate class.

Bayesian Classifier

- In many applications, the relationship between the attributes set and the class variable is non-deterministic.
  - In other words, a test cannot be classified to a class label with certainty.
  - In such a situation, the classification can be achieved probabilistically.
- The Bayesian classifier is an approach for modelling probabilistic relationships between the attribute set and the class variable.
- More precisely, Bayesian classifier use Bayes' Theorem of Probability for classification.
- Before going to discuss the Bayesian classifier, we should have a quick look at the Theory of Probability and then Bayes' Theorem.

Bayes’ Theorem of Probability

Simple Probability

Definition 8.2: **Simple Probability**

If there are n elementary events associated with a random experiment and m of n of them are favorable to an event A, then the probability of happening or occurrence of A is

Simple Probability

- Suppose, A and B are any two events and *P(A)*, *P(B)* denote the probabilities that the events *A* and *B* will occur, respectively.

- **Mutually Exclusive Events:**
  - Two events are mutually exclusive, if the occurrence of one precludes the occurrence of the other.

  Example: Tossing a coin (two events)

  Tossing a ludo cube (Six events)

💡 Can you give an example, so that two events are not mutually exclusive?

Hint: Tossing two identical coins, Weather (sunny, foggy, warm)

Simple Probability

- **Independent events:** Two events are independent if occurrences of one does not alter the occurrence of other.

Example: Tossing both coin and ludo cube together.

(How many events are here?)

💡 Can you give an example, where an event is dependent on one or more other events(s)?

Hint: Receiving a message (A) through a communication channel (B) over a computer (C), rain and train.

Joint Probability

Definition 8.3: **Joint Probability**

If *P(A)* and *P(B)* are the probability of two events, then

If *A* and *B* are mutually exclusive, then
If *A* and *B* are independent events, then

Thus, for mutually exclusive events

Conditional Probability

Definition 8.2: **Conditional Probability**

If events are dependent, then their probability is expressed by conditional probability. The probability that *A* occurs given that *B* is denoted by .

Suppose, *A* and *B* are two events associated with a random experiment. The probability of *A* under the condition that *B* has already occurred and is given by

Conditional Probability

Corollary 8.1: **Conditional Probability**

or

For three events A, B and C

For n events A_1, A_2, …, A_n and if all events are mutually independent to each other

Note:

if events are **mutually exclusive**

if A and B are **independent**

otherwise,

Conditional Probability

- Generalization of Conditional Probability:

$$\because \ P(A) = P(B)$$

By the law of total probability : P(B) =

Conditional Probability

In general,

Total Probability

Definition 8.3: Total Probability

Let be *n* mutually exclusive and exhaustive events associated with a random experiment. If *A* is any event which occurs with , then

Total Probability: An Example

Example 8.3

A bag contains 4 red and 3 black balls. A second bag contains 2 red and 4 black balls. One bag is selected at random. From the selected bag, one ball is drawn. What is the probability that the ball drawn is red?

This problem can be answered using the concept of Total Probability

Selecting bag *I*

Selecting bag *II*

A = Drawing the red ball

Thus,

where, = Probability of drawing red ball when first bag has been chosen

and = Probability of drawing red ball when second bag has been chosen

Reverse Probability

Example 8.3:

A bag (Bag I) contains 4 red and 3 black balls. A second bag (Bag II) contains 2 red and 4 black balls. You have chosen one ball at random. It is found as red ball. What is the probability that the ball is chosen from Bag I?

Here,

Selecting bag *I*

Selecting bag *II*

A = Drawing the red ball

We are to determine P(|A). Such a problem can be solved using Bayes' theorem of probability.

Bayes' Theorem

Theorem 8.4: **Bayes' Theorem**

Let be *n* mutually exclusive and exhaustive events associated with a random experiment. If *A* is any event which occurs with , then

Prior and Posterior Probabilities

- P(A) and P(B) are called prior probabilities
- P(A|B), P(B|A) are called posterior probabilities

Example 8.6: Prior versus Posterior Probabilities

- This table shows that the event Y has two outcomes namely A and B, which is dependent on another event X with various outcomes like and .
- **Case1:** Suppose, we don't have any information of the event A. Then, from the given sample space, we can calculate $P(Y = A) =$ = 0.5
  .
- **Case2:** Now, suppose, we want to calculate $P(X = |Y = A) = =$ 0.4 .

The later is the conditional or posterior probability, where as the former is the prior probability.

X	Y
	A
	A
	B
	A
	B
	A
	B
	B
	B
	A

Naïve Bayesian Classifier

- Suppose, *Y* is a class variable and *X* = is a set of attributes,

  with instance of *Y*.

INPUT (X)	CLASS(Y)

- The classification problem, then can be expressed as the class-conditional probability

Naïve Bayesian Classifier

- Naïve Bayesian classifier calculate this posterior probability using Bayes' theorem, which is as follows.

- From Bayes' theorem on conditional probability, we have

where,

(Y)

Note:

- is called the evidence (also the total probability) and it is a constant.

- The probability *P(Y|X)* (also called class conditional probability) is therefore proportional to *P(X|Y)*.

- Thus, *P(Y|X)* can be taken as a measure of *Y* given that *X*.

$P(Y|X)$

Naïve Bayesian Classifier

- Suppose, for a given instance of *X* (say *x* = () and ….. .
- There are any two class conditional probabilities namely *P(Y|X=x)* and *P(YX=x)*.
- If *P(YX=x)* > *P(YX=x)*, then we say that is more stronger than for the instance *X* = *x*.
- The strongest is the classification for the instance *X* = *x*.

Naïve Bayesian Classifier

- **Example:** With reference to the Air Traffic Dataset mentioned earlier, let us tabulate all the posterior and prior probabilities as shown below.

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Day	Weekday	9/14 = 0.64	½ = 0.5	3/3 = 1	0/1 = 0
Season					
	Winter	2/14 = 0.14	2/2 = 1	2/3 = 0.67	0/1 = 0

Naïve Bayesian Classifier

		Class			
Attribute		On Time	Late	Very Late	Cancelled
Fog					
	High	4/14 = 0.29	1/2 = 0.5	1/3 = 0.33	1/1 = 1
Rain					
	Heavy	1/14 = 0.07	1/2 = 0.5	2/3 = 0.67	1/1 = 1
Prior Probability					

Naïve Bayesian Classifier

Instance:

Week Day	Winter	High	Heavy	???

Case1: Class = On Time : 0.70 × 0.64 × 0.14 × 0.29 × 0.07 = 0.0013

Case2: Class = Late : 0.10 × 0.50 × 1.0 × 0.50 × 0.50 = 0.0125

Case3: Class = Very Late : 0.15 × 1.0 × 0.67 × 0.33 × 0.67 = 0.0222

Case4: Class = Cancelled : 0.05 × 0.0 × 0.0 × 1.0 × 1.0 = 0.0000

Case3 is the strongest; Hence correct classification is **Very Late**

Naïve Bayesian Classifier

Algorithm: Naïve Bayesian Classification

Input: Given a set of *k* mutually exclusive and exhaustive classes *C* = , which have prior probabilities $P(C_1)$, $P(C_2)$,….. $P(C_k)$.

There are *n*-attribute set *A* = which for a given instance have values = , = ,….., =

Step: For each , calculate the class condition probabilities, *i* = 1,2,…..,*k*

Output: is the classification

Note: , because they are not probabilities rather proportion values (to posterior probabilities)

Naïve Bayesian Classifier

Pros and Cons

- The Naïve Bayes’ approach is a very popular one, which often works well.
- However, it has a number of potential problems
  - It relies on all attributes being categorical.
  - If the data is less, then it estimates poorly.

Naïve Bayesian Classifier

Approach to overcome the limitations in Naïve Bayesian Classification

- Estimating the posterior probabilities for continuous attributes

  - In real life situation, all attributes are not necessarily be categorical, In fact, there is a mix of both categorical and continuous attributes.

  - In the following, we discuss the schemes to deal with continuous attributes in Bayesian classifier.

1. We can discretize each continuous attributes and then replace the continuous values with its corresponding discrete intervals.

2. We can assume a certain form of probability distribution for the continuous variable and estimate the parameters of the distribution using the training data. A Gaussian distribution is usually chosen to represent the posterior probabilities for continuous attributes. A general form of Gaussian distribution will look like

   where, denote mean and variance, respectively.

Naïve Bayesian Classifier

For each class C_i, the posterior probabilities for attribute A_j (it is the numeric attribute) can be calculated following Gaussian normal distribution as follows.

Here, the parameter can be calculated based on the sample mean of attribute value of for the training records that belong to the class .

Similarly, can be estimated from the calculation of variance of such training records.

Naïve Bayesian Classifier

M-estimate of Conditional Probability

- The M-estimation is to deal with the potential problem of Naïve Bayesian Classifier when training data size is too poor.
    - If the posterior probability for one of the attribute is zero, then the overall class-conditional probability for the class vanishes.
    - In other words, if training data do not cover many of the attribute values, then we may not be able to classify some of the test records.
- This problem can be addressed by using the M-estimate approach.

M-estimate Approach

- M-estimate approach can be stated as follows

where, *n* = total number of instances from class

= number of training examples from class that take the value

m = it is a parameter known as the equivalent sample size, and

p = is a user specified parameter.

Note:

If *n* = 0, that is, if there is no training set available, then = *p*,

so, this is a different value, in absence of sample value.

A Practice Example

Example 8.4

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data instance
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = fair)

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

A Practice Example

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 × 0.444 × 0.667 × 0.667 = 0.044
P(X|buys_computer = “no”) = 0.6 × 0.4 × 0.2 × 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Thank You!

Brain Computer Interaction

Feature Translation – Decision Trees

<u>Course Instructors</u>

Dr. Annushree Bablani

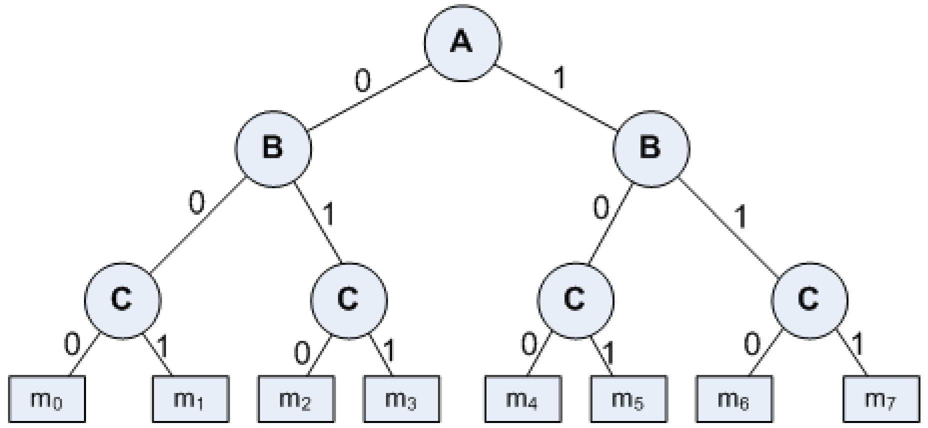
Acknowledgements: Dr. Sreeja S R

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems

Example 8.1: Binary Decision Tree

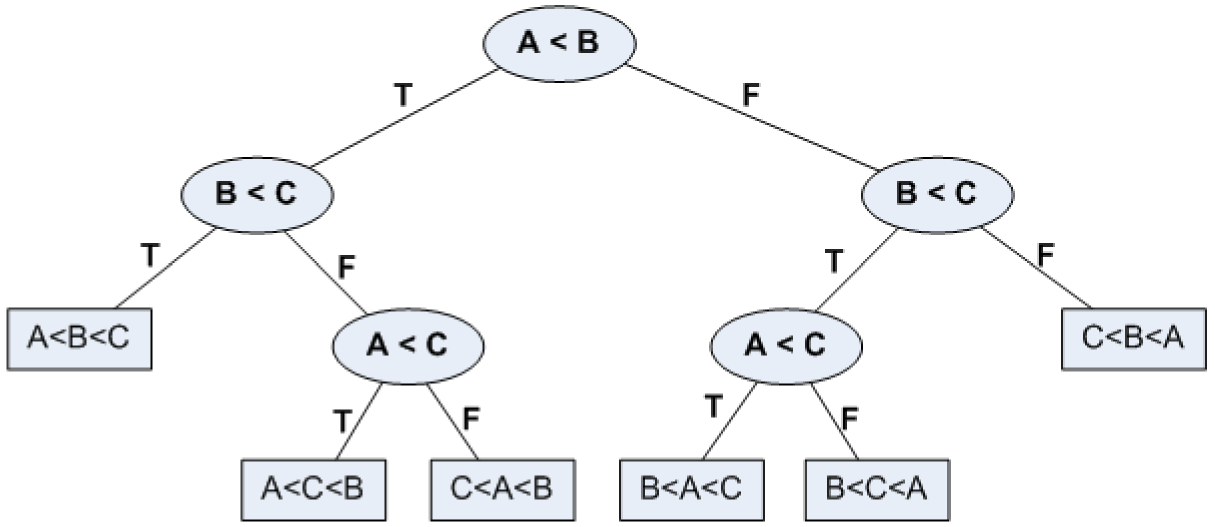
A	*B*	*C*	*f*
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



Basic Concept

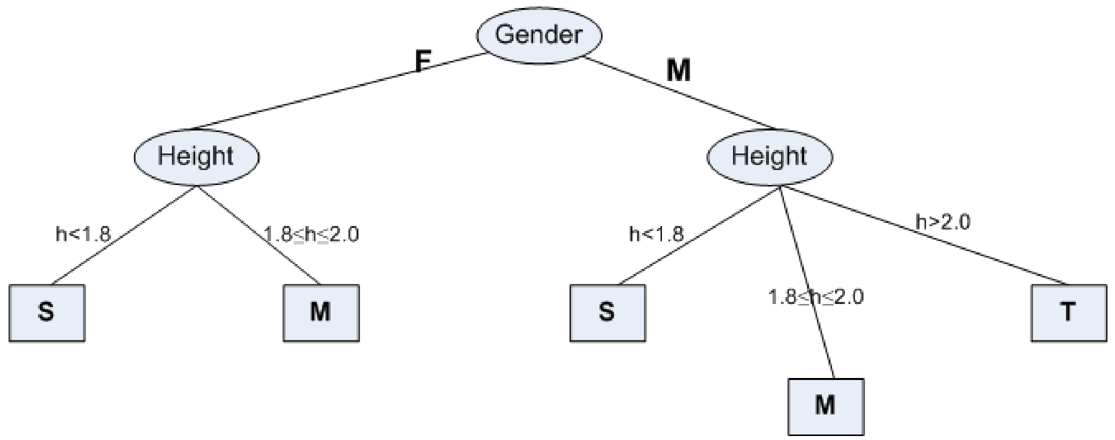
- In Example 18.1, we have considered a decision tree where values of any attribute if binary only. Decision tree is also possible where attributes are of continuous data type

Example 8.2: Decision Tree with numeric data



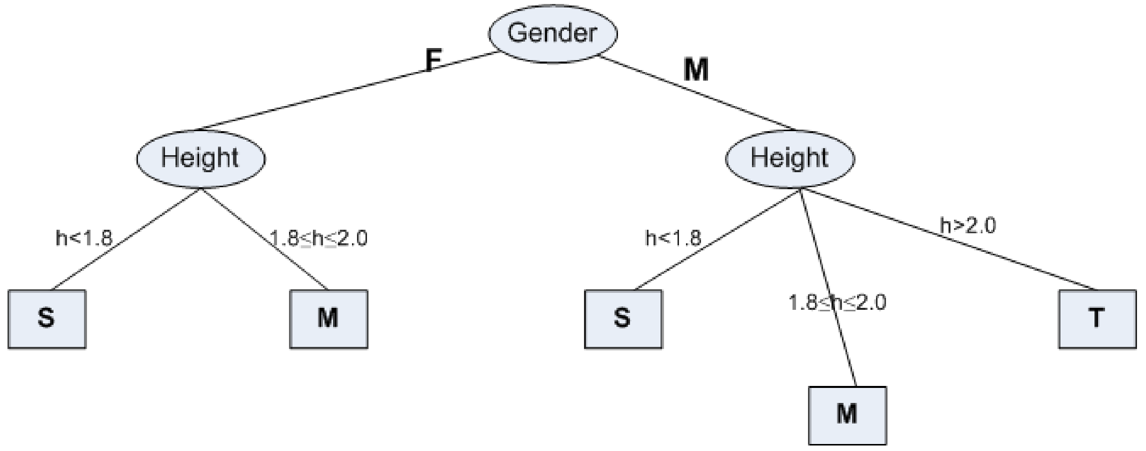
Some Characteristics

- Decision tree may be *n*-ary, *n* ≥ 2.
- There is a special node called root node.
- All nodes drawn with circle (ellipse) are called internal nodes.
- All nodes drawn with rectangle boxes are called terminal nodes or leaf nodes.
- Edges of a node represent the outcome for a value of the node.
- In a path, a node with same label is never repeated.
- Decision tree is not unique, as different ordering of internal nodes can give different decision tree.



Decision Tree and Classification Task

- Decision tree helps us to classify data.
  - Internal nodes are some attribute
  - Edges are the values of attributes
  - External nodes are the outcome of classification



- Such a classification is, in fact, made by posing questions starting from the root node to each terminal node.

Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	**Mammal**
Python	Cold	scales	no	no	no	no	yes	**Reptile**
Salmon	Cold	scales	no	yes	no	no	no	**Fish**
Whale	Warm	hair	yes	yes	no	no	no	**Mammal**
Frog	Cold	none	no	semi	no	yes	yes	**Amphibian**
Komodo	Cold	scales	no	no	no	yes	no	**Reptile**
Bat	Warm	hair	yes	no	yes	yes	yes	**Mammal**
Pigeon	Warm	feathers	no	no	yes	yes	no	**Bird**
Cat	Warm	fur	yes	no	no	yes	no	**Mammal**
Leopard	Cold	scales	yes	yes	no	no	no	**Fish**
Turtle	Cold	scales	no	semi	no	yes	no	**Reptile**
Penguin	Warm	feathers	no	semi	no	yes	no	**Bird**
Porcupine	Warm	quills	yes	no	no	yes	yes	**Mammal**
Eel	Cold	scales	no	yes	no	no	no	**Fish**
Salamander	Cold	none	no	semi	no	yes	yes	**Amphibian**

What are the class label of Dragon and Shark?

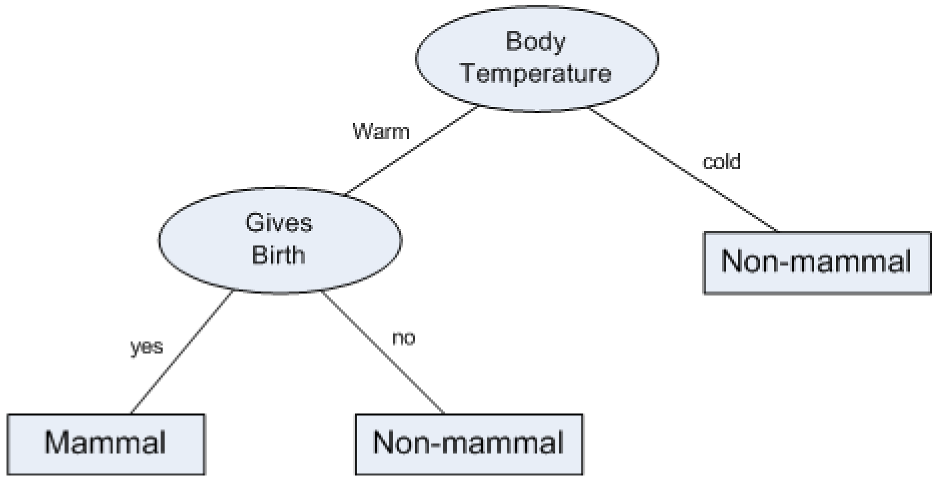
Decision Tree and Classification Task

Example 8.3 : Vertebrate Classification

- Suppose, a new species is discovered as follows.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Gila Monster	cold	scale	no	no	no	yes	yes	?

- Decision Tree that can be inducted based on the data (in Example 8.3) is as follows.



Decision Tree and Classification Task

- The above Example illustrates how we can solve a classification problem by asking a series of question about the attributes.
  - Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- The series of questions and their answers can be organized in the form of a decision tree
  - As a hierarchical structure consisting of nodes and edges
- Once a decision tree is built, it is applied to any test to classify it.

Definition of Decision Tree

Definition 1: **Decision Tree**

Given a database *D* = here denotes a tuple, which is defined by a set of attribute set of classes *C* = .

A decision tree ***T*** is a tree associated with *D* that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

Building Decision Tree

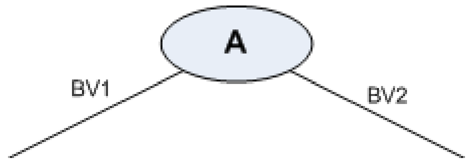
- In principle, there are exponentially many decision tree that can be constructed from a given database (also called training data).
  - Some of the tree may not be optimum
  - Some of them may give inaccurate result

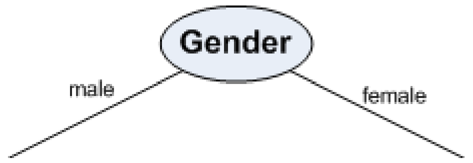
- Two approaches are known
  - **Greedy strategy**
    - A top-down recursive divide-and-conquer
  - **Modification of greedy strategy**
    - ID3
    - C4.5
    - CART, etc.

Node Splitting in **BuildDT** Algorithm

- BuildDT algorithm must provides a method for expressing an attribute test condition and corresponding outcome for different attribute type

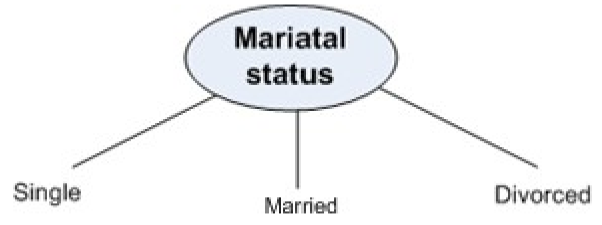
- **Case: Binary attribute**
  - This is the simplest case of node splitting
  - The test condition for a binary attribute generates only two outcomes



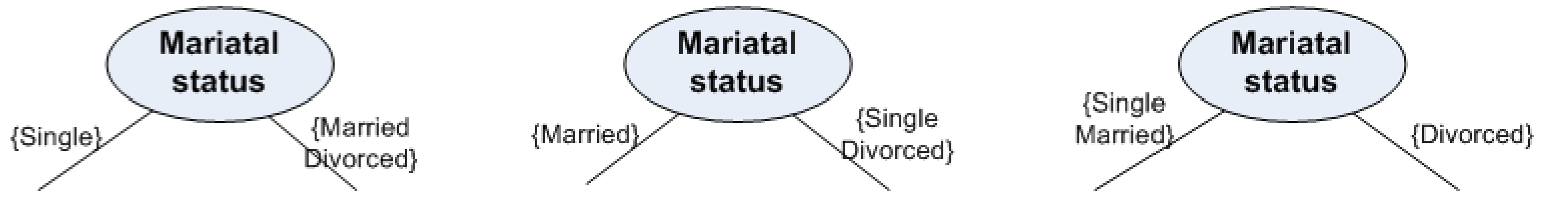


Node Splitting in **BuildDT** Algorithm

- **Case: Nominal attribute**
  - Since a nominal attribute can have many values, its test condition can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Muti-way split: Outcome depends on the number of distinct values for the corresponding attribute

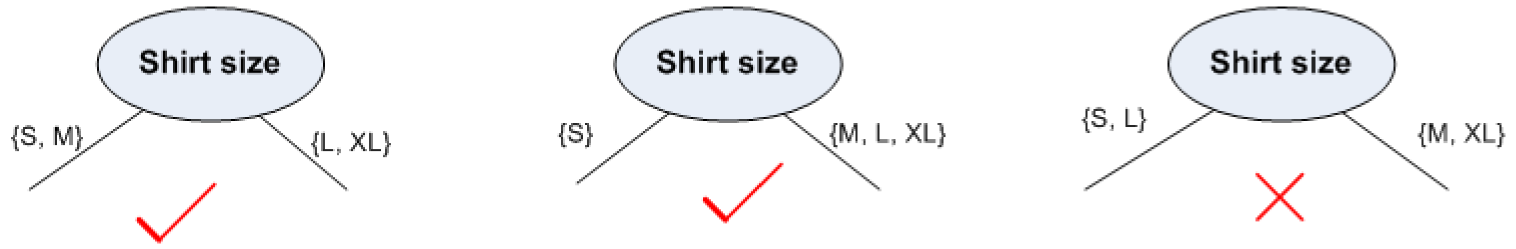


  - Binary splitting by grouping attribute values



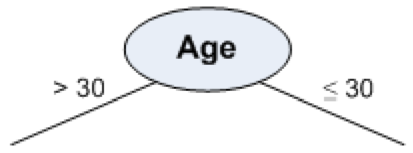
Node Splitting in **BuildDT** Algorithm

- **Case: Ordinal attribute**
  - It also can be expressed in two ways:
    - A multi-way split
    - A binary split
  - Multi-way split: It is same as in the case of nominal attribute
  - Binary splitting attribute values should be grouped maintaining the order property of the attribute values



Node Splitting in **BuildDT** Algorithm

- **Case: Numerical attribute**
  - For numeric attribute (with discrete or continuous values), a test condition can be expressed as a comparison set
    - Binary outcome: $A > v$ or $A \le v$
      - In this case, decision tree induction must consider all possible split positions
    - Range query : $v_i \le A < v_{i+1}$ for i = 1, 2, …, q (if q number of ranges are chosen)
      - Here, q should be decided a priori
  - For a numeric attribute, decision tree induction is a combinatorial optimization problem



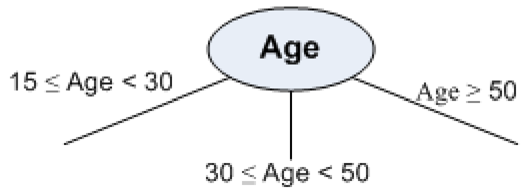


Illustration : **BuildDT** Algorithm

Example 8.4: Illustration of BuildDT Algorithm

- Consider a training data set as shown.

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

Attributes:

Gender = {Male(M), Female (F)} // Binary attribute
Height = {1.5, …, 2.5} // Continuous attribute

Class = {Short (S), Medium (M), Tall (T)}

Given a person, we are to test in which class s/he belongs

Illustration : **BuildDT** Algorithm

- To built a decision tree, we can select an attribute in two different orderings: <Gender, Height> or <Height, Gender>
- Further, for each ordering, we can choose different ways of splitting
- Different instances are shown in the following.
- **Approach 1 : <Gender, Height>**

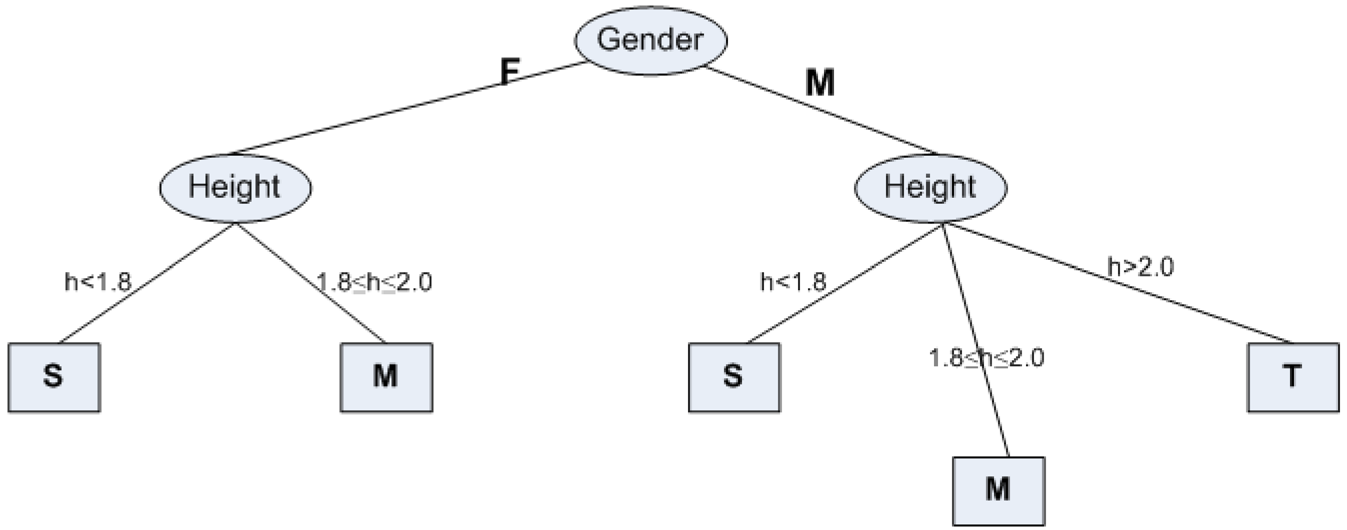
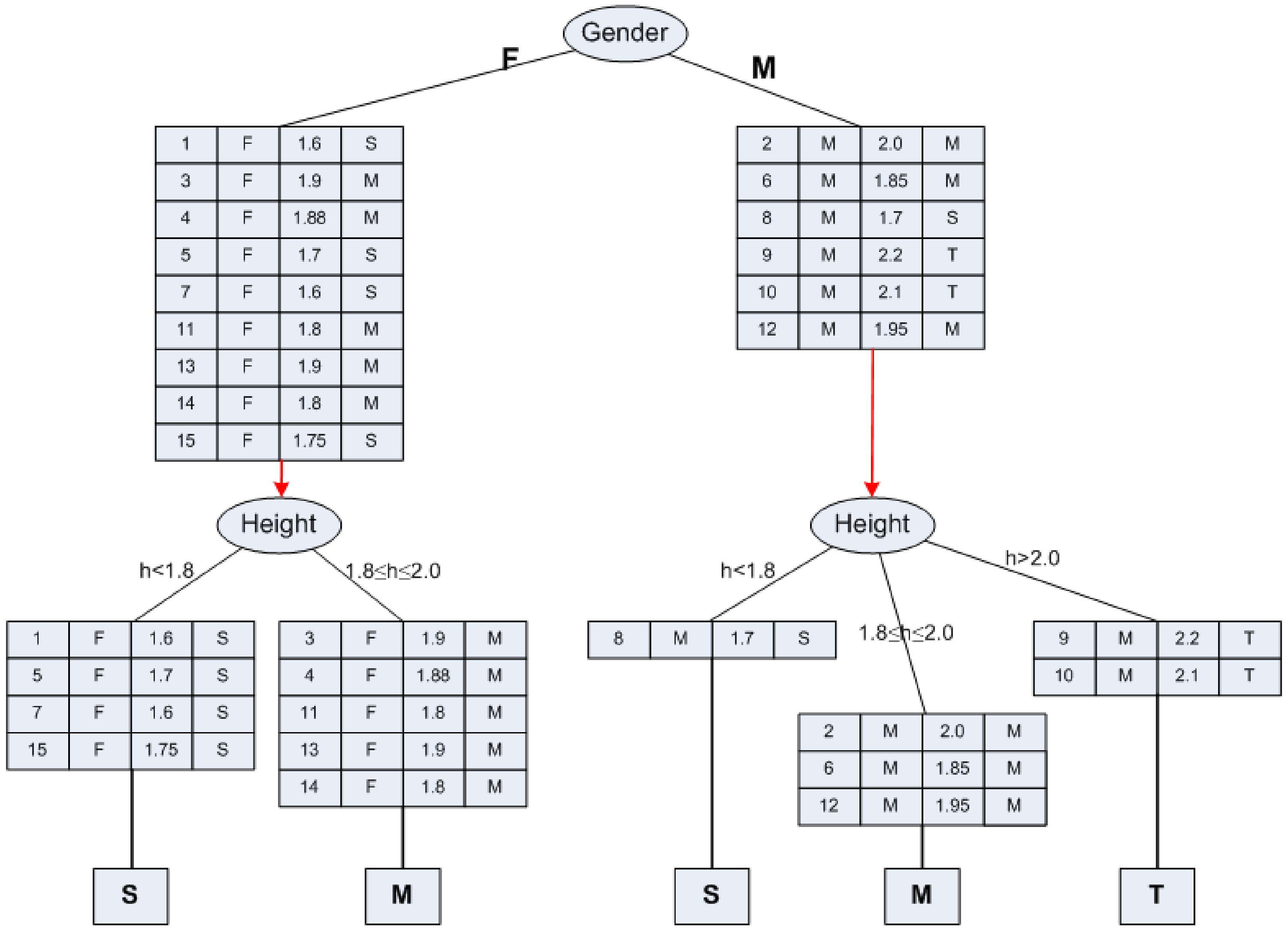


Illustration : **BuildDT** Algorithm



1	F	1.6	S
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
7	F	1.6	S
11	F	1.8	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

2	M	2.0	M
6	M	1.85	M
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
12	M	1.95	M

1	F	1.6	S
5	F	1.7	S
7	F	1.6	S
15	F	1.75	S

3	F	1.9	M
4	F	1.88	M
11	F	1.8	M
13	F	1.9	M
14	F	1.8	M

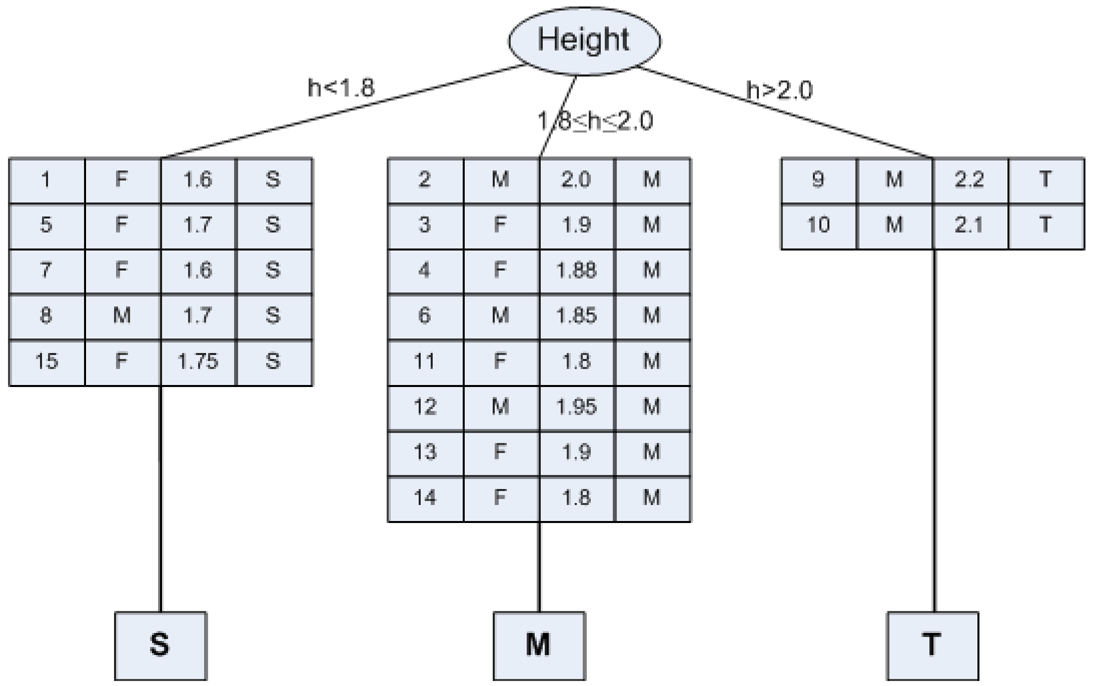
8	M	1.7	S

2	M	2.0	M
6	M	1.85	M
12	M	1.95	M

9	M	2.2	T
10	M	2.1	T

Illustration : **BuildDT** Algorithm

- **Approach 2 : <Height, Gender>**



1	F	1.6	S
5	F	1.7	S
7	F	1.6	S
8	M	1.7	S
15	F	1.75	S

2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
6	M	1.85	M
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M

9	M	2.2	T
10	M	2.1	T

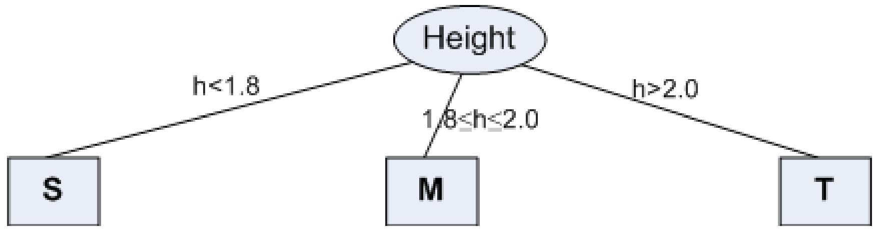


Illustration : **BuildDT** Algorithm

Example 8.5: Illustration of BuildDT Algorithm

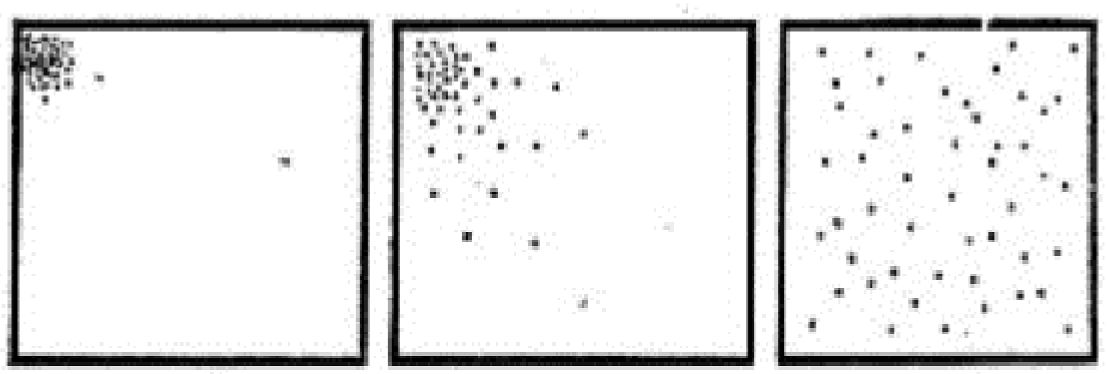
- Consider an anonymous database as shown.

A1	A2	A3	A4	Class
a11	a21	a31	a41	C1
a12	a21	a31	a42	C1
a11	a21	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a11	a22	a32	a41	C2
a11	a22	a31	a42	C1
a11	a21	a32	a42	C2
a11	a22	a32	a41	C2
a12	a22	a31	a41	C1
a12	a22	a31	a42	C1

- Is there any "clue" that enables to select the "best" attribute first?
- Suppose, following are two attempts:
  - A1→A2→A3→A4 [naïve]
  - A3→A2→A4→A1 [Random]
- Draw the decision trees in the above-mentioned two cases.
- Are the trees different to classify any test data?
- If any other sample data is added into the database, is that likely to alter the decision tree already obtained?

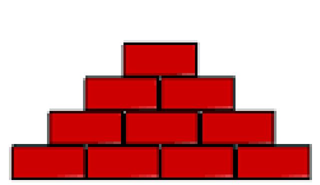
Concept of Entropy

Concept of Entropy

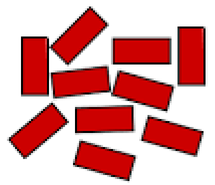


If a point represents a gas molecule, then which system has the more entropy?

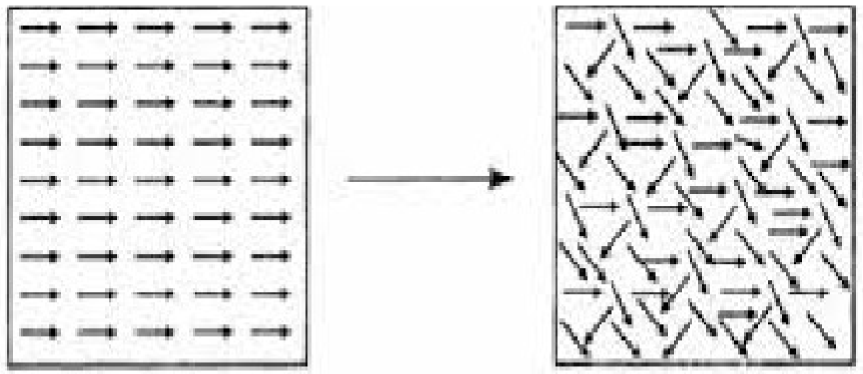
How to measure? ?



More **ordered** less **entropy**



Less ordered **higher** entropy



More organized or **ordered** (less **probable**)

Less organized or disordered (**more** probable)

An Open Challenge!

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	89	99	56	91
10IM30013	95	98	55	93
12CE31005	98	96	58	97
12EC35015	93	95	54	99
12GG2005	90	91	53	98
12MI33006	91	93	57	97
13AG36001	96	94	58	95
13EE10009	92	96	56	96
13MA20012	88	98	59	96
14CS30017	94	90	60	94
14ME10067	90	92	58	95
14MT10038	99	89	55	93

Roll No.	Assignment	Project	Mid-Sem	End-Sem
12BT3FP06	19	59	16	71
10IM30013	37	38	25	83
12CE31005	38	16	48	97
12EC35015	23	95	54	19
12GG2005	40	71	43	28
12MI33006	61	93	47	97
13AG36001	26	64	48	75
13EE10009	92	46	56	56
13MA20012	88	58	59	66
14CS30017	74	20	60	44
14ME10067	50	42	38	35
14MT10038	29	69	25	33

Two sheets showing the tabulation of marks obtained in a course are shown.

Which tabulation of marks shows the “good” performance of the class?
How you can measure the same?

Entropy and its Meaning

- Entropy is an important concept used in Physics in the context of heat and thereby uncertainty of the states of a matter.

- At a later stage, with the growth of Information Technology, entropy becomes an important concept in Information Theory.

- To deal with the classification job, entropy is an important concept, which is considered as

  - an information-theoretic measure of the "uncertainty" contained in a training data

    - due to the presence of more than one classes.

Entropy in Information Theory

- The entropy concept in information theory first time coined by Claude Shannon (1850).

- The first time it was used to measure the “information content” in messages.

- According to his concept of entropy, presently entropy is widely being used as a way of representing messages for efficient transmission by Telecommunication Systems.

Measure of Information Content

- People, in general, are information hungry!
- Everybody wants to acquire information (from newspaper, library, nature, fellows, etc.)
  - Think how a crime detector do it to know about the crime from crime spot and criminal(s).
  - Kids annoyed their parents asking questions.
- In fact, fundamental thing is that we gather information asking questions (and decision tree induction is no exception).
- We may note that information gathering may be with certainty or uncertainty.

Measure of Information Content

Example 8.6

a) Guessing a birthday of your classmate

It is with uncertainty ~

Whereas guessing the day of his/her birthday is .

This uncertainty, we may say varies between 0 to 1, both inclusive.

b) As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

• Does sun rises in the East?	(answer is with 0 uncertainty)
• Will mother give birth to male baby?	(answer is with ½ uncertainty)
• Is there a planet like earth in the galaxy?	(answer is with an extreme uncertainty)

Entropy Calculation

- If there are *m* objects with frequencies , ……., , then the average number of bits (i.e. questions) that need to be examined a value, that is, entropy is the frequency of occurrence of the value multiplied by the number of bits that need to be determined, summed up values of from *1* to *m*.

Theorem: Entropy calculation

If p_i denotes the frequencies of occurrences of *m* distinct objects, then the entropy *E* is

Note:

- If all are equally likely, then and ; it is the special case.

Entropy of a Training Set

- If there are *k* classes , ……., and for denotes the number of occurrences of classes divided by the total number of instances (i.e., the frequency of occurrence of) in the training set, then entropy of the training set is denoted by

Here, *E* is measured in "bits" of information.

Note:

- The above formula should be summed over the non-empty classes only, that is, classes for which
- *E* is always a positive quantity
- *E* takes it's minimum value (zero) if and only if all the instances have the same class (i.e., the training set with only **one** non-empty class, for which the probability 1).
- Entropy takes its maximum value when the instances are equally distributed among *k* possible classes. In this case, the maximum value of *E* is .

Entropy of a Training Set

Example 18.10: OPTH dataset

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

A coded forms for all values of attributes are used to avoid the cluttering in the table.

Entropy of a training set

Specification of the attributes are as follows.

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Class: 1: Contact Lens 2:Normal glass 3: Nothing

In the OPTH database, there are 3 classes and 4 instances with class 1, 5 instances with class 2 and 15 instances with class 3. Hence, entropy *E* of the database is:

Note:

- The entropy of a training set implies the number of yes/no questions, on the average, needed to determine an unknown test to be classified.

- It is very crucial to decide the series of questions about the value of a set of attribute, which collectively determine the classification. Sometimes it may take one question, sometimes many more.

- Decision tree induction helps us to ask such a series of questions. In other words, we can utilize entropy concept to build a better decision tree.

How entropy can be used to build a decision tree ?

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.

- Different algorithms have been proposed to take a good control over
  1. Choosing the best attribute to be splitted, and
  2. Splitting criteria

- Several algorithms have been proposed for the above tasks. In this lecture, we shall limit our discussions into three important of them
  - **ID3**
  - **C 4.5**
  - **CART**

Algorithm ID3

ID3: Decision Tree Induction Algorithms

- Quinlan [1984] introduced the ID3, a popular short form of **I**terative **D**ichotomizer 3 for decision trees from a set of training data.

- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.

- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

Algorithm ID3

- In ID3, entropy is used to measure how informative a node is.
  - It is observed that splitting on any attribute has the property that average entropy of the resulting training subsets will be less than or equal to that of the previous training set.
- ID3 algorithm defines a measurement of a splitting called **Information Gain** to determine the goodness of a split.
  - The attribute with the largest value of information gain is chosen as the splitting attribute and
  - it partitions into a number of smaller training sets based on the distinct values of attribute under split.

Defining Information Gain

- We consider the following symbols and terminologies to define information gain, which is denoted as α.
- *D* denotes the training set at any instant
- $|D|$ denotes the size of the training set *D*
- $E(D)$ denotes the entropy of the training set *D*
- The entropy of the training set *D*

$$E(D) = -$$

  - where the training set *D* has , , … , , the *k* number of distinct classes and
  - , 0< is the probability that an arbitrary tuple in *D* belongs to class $(i = 1, 2, \ldots , k)$.

Defining Information Gain

Definition 2: **Weighted Entropy**

The weighted entropy denoted as $E_A(D)$ for all partitions of D with respect to A is given by:

$= E()$

Here, the term denotes the weight of the *j*-th training set.

More meaningfully, $E_A(D)$ is the expected information required to classify a tuple from D based on the splitting of A.

Defining Information Gain

- Our objective is to take *A* on splitting to produce an exact classification (also called pure), that is, all tuples belong to one class.
- However, it is quite likely that the partitions is impure, that is, they contain tuples from two or more classes.
- In that sense, is a measure of impurities (or purity). A lesser value of implying more power the partitions are.

Definition 3: Information Gain

Information gain, of the training set *D* splitting on the attribute *A* is given by

$= E(D) -$

In other words, gives us an estimation how much would be gained by splitting on *A*. The attribute *A* with the highest value of should be chosen as the splitting attribute for *D*.

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

- Let us refer to the OPTH database discussed earlier.

- Splitting on **Age** at the root level, it would give three subsets and as shown in the tables in the following three slides.

- The entropy and of training sets and and corresponding weighted entropy and are also shown alongside.

- The Information gain is then can be calculated as **0.0394**.

- Recall that entropy of OPTH data set, we have calculated as *E(OPTH)* = **1.3261**

  (see Slide #16)

Information Gain Calculation

Example 8.11 : Information gain on splitting OPTH

Training set: (Age = 1)

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

() ()

() = **1.5**

$\times$ 1.5 = **0.5000**

Calculating Information Gain

Training set: (Age = 2)

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

() () ()

$= \mathbf{1.2988}$

$\times\ 1.2988 = \mathbf{0.4329}$

Calculating Information Gain

Training set: (Age = 3)

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

() ()

$() = \mathbf{1.0613}$

$\times\ 1.0613 = \mathbf{0.3504}$

$= 1.3261 - (0.5000 + 0.4329 + 0.3504) = \mathbf{0.0394}$

Information Gains for Different Attributes

- In the same way, we can calculate the information gains, when splitting the OPTH database on Eye-sight, Astigmatic and Use Type. The results are summarized below.
- Splitting attribute: Age
- Splitting attribute: Eye-sight
- Splitting attribute: Astigmatic

  770
- Splitting attribute: Use Type

  5488

Decision Tree Induction : ID3 Way

- The ID3 strategy of attribute selection is to choose to split on the attribute that gives the greatest reduction in the weighted average entropy
  - The one that maximizes the value of information gain
- In the example with OPTH database, the larger values of information gain is 5488
  - Hence, the attribute should be chosen for splitting is “Use Type”.
- The process of splitting on nodes is repeated for each branch of the evolving decision tree, and the final tree, which would look like is shown in the following slide and calculation is left for practice.

Decision Tree Induction : ID3 Way

$E(OPTH) = 1.3261$

$OPTH$

Age ✗ $\alpha = 0.0394$

Eye-sight ✗ $\alpha = 0.395$

Use Type ✓ $\alpha = 0.5488$

Astigmatic ✗ $\alpha = 0.3770$

$D1$ $Frequent(1)$

E(D¿¿1)=?¿

Age	Eye	Ast	Use	Class
1	1	1	1	3
1	1	2	1	3
1	2	1	1	3
1	2	2	1	3
2	1	1	1	3
2	2	1	1	3
2	2	2	1	3
3	1	1	1	3
3	1	2	1	3
3	2	1	1	3
3	2	2	1	3

Age ? $\alpha = ?$

Eye-sight ? $\alpha = ?$

Astigmatic ? $\alpha = ?$

$D2$ $Less(2)$

E(D¿¿2)=?¿

Age	Eye	Ast	Use	Class
1	1	1	2	2
1	1	2	2	1
1	2	1	2	2
1	2	2	2	1
2	1	1	2	2
2	1	2	2	1
2	2	1	2	2
3	1	1	2	3
3	1	2	2	3
3	2	1	2	2
3	2	2	2	3

Age ? $\alpha = ?$

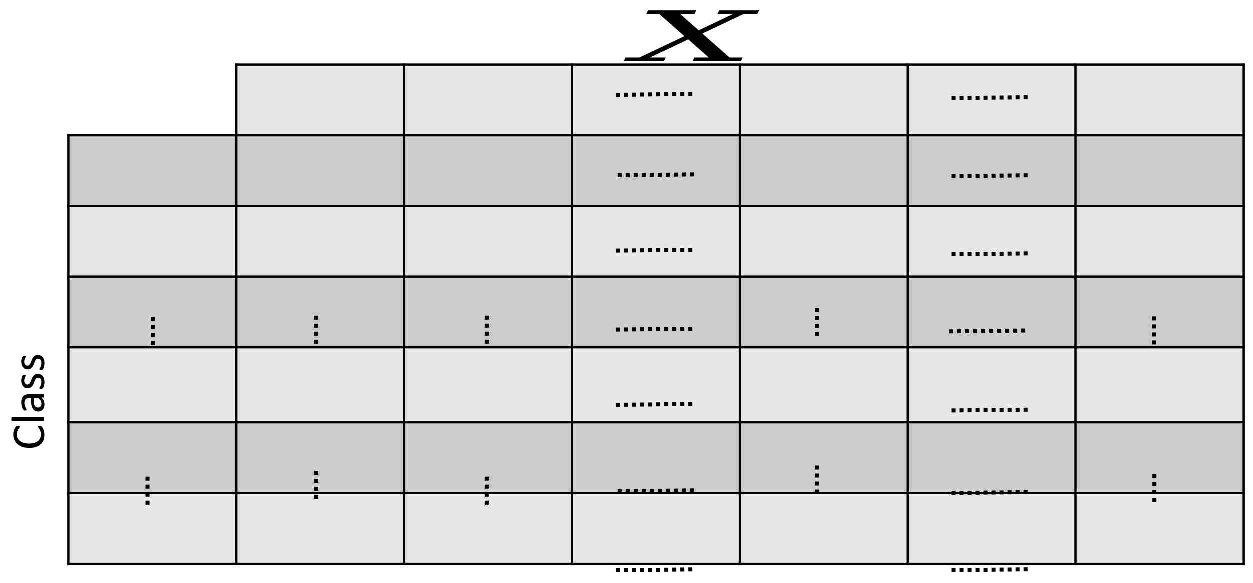
Eye-sight ? $\alpha = ?$

Astigmatic ? $\alpha = ?$

Frequency Table : Calculating α

- Calculation of entropy for each table and hence information gain for a particular split appears tedious (at least manually)!
- As an alternative, we discuss **a short-cut method** of doing the same using a special data structure called **Frequency Table.**
- **Frequency Table**: Suppose, denotes an attribute with attribute values in it. For a given database , there are a set of say . Given this, a frequency table will look like as follows.

Frequency Table : Calculating α



- Number of rows = Number of classes
- Number of columns = Number of attribute values
- = Frequency of for class

Assume that , the number of total instances of .

Calculation of α using Frequency Table

Example 8.12 : OTPH Dataset

With reference to OPTH dataset, and for the attribute Age, the frequency table would look like

	Age=1	Age=2	Age=3	Row Sum
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column Sum	8	8	8	24

N=24

Calculation of α using Frequency Table

- The weighted average entropy then can be calculated from the frequency table following the
  - Calculate for all
    (Entry Sum) and
  - Calculate for all
    (Column Sum) in the row of column sum
  - Calculate

Example 8.13: OTPH Dataset
For the frequency table in **Example 18.12**, we have

Limiting Values of Information Gain

- The Information gain metric used in ID3 always should be positive or zero.

- It is always positive value because information is always gained (i.e., purity is improved) by splitting on an attribute.

- On the other hand, when a training set is such that if there are classes, and the entropy of training set takes the largest value i.e., (this occurs when the classes are balanced), then the information gain will be zero.

Limiting Values of Information Gain

Example 8.14: Limiting values of Information gain

Consider a training set shown below.

Data set *Table A*

X	Y	Clas s
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

X *Table X*

	1	2	3	4
A	1	1	1	1
B	1	1	1	1
	2	2	2	2

Frequency table of X

Y *Table Y*

	1	2
A	2	2
B	2	2
	4	4

Frequency table of Y

Limiting values of Information Gain

- Entropy of Table A is

(The maximum entropy).

- In this example, whichever attribute is chosen for splitting, each of the branches will also be balanced thus each with maximum entropy.
- In other words, information gain in both cases (i.e., splitting on *X* as well as *Y*) will be zero.

Note:

- The absence of information gain does not imply that there is no profit for splitting on the attribute.
- Even if it is chosen for splitting, ultimately it will lead to a final decision tree with the branches terminated by a leaf node and thus having an entropy of zero.
- Information gain can never be a negative value.

Reference

- The detail material related to this lecture can be found in

Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?

Brain Computer Interaction

Feature Translation – Decision Tree Induction – CART & C4.5

<u>Course Instructors</u>

Dr. Annushree Bablani

Acknowledgements: Dr. Sreeja S R

Algorithm CART

CART Algorithm

- It is observed that information gain measure used in ID3 is biased towards test with many outcomes, that is, it prefers to select attributes having a large number of values.

- L. Breiman, J. Friedman, R. Olshen and C. Stone in 1984 proposed an algorithm to build a binary decision tree also called CART decision tree.
  - CART stands for **Classification and Regression Tree**
  - In fact, invented independently at the same time as ID3 (1984).
  - ID3 and CART are two cornerstone algorithms spawned a flurry of work on decision tree induction.

- CART is a technique that generates a **binary decision tree;** That is, unlike ID3, in CART, for each node only two children is created.

- ID3 uses Information gain as a measure to select the best attribute to be splitted, whereas CART do the same but using another measurement called **Gini index** . It is also known as **Gini Index of Diversity** and is denote as .

Gini Index of Diversity

Definition 20.1: **Gini Index**

Suppose, D is a training set with size $|D|$ and be the set of k classifications and be any attribute with m different values of it. Like entropy measure in ID3, CART proposes Gini Index (denoted by G) as the measure of impurity of D. It can be defined as follows.

where is the probability that a tuple in D belongs to class and can be estimated as

where denotes the number of tuples in D with class .

Gini Index of Diversity

Note

- measures the "impurity" of data set *D*.
- The smallest value of is zero
  - which it takes when all the classifications are same.
- It takes its largest value
  - when the classes are evenly distributed between the tuples, that is the frequency of each class is .

Gini Index of Diversity

Definition 20.2: **Gini Index of Diversity**

Suppose, a binary partition on *A* splits *D* into and , then the weighted average Gini Index of splitting denoted by is given by

This binary partition of *D* reduces the impurity and the reduction in impurity is measured by

Gini Index of Diversity and CART

- This is called the Gini Index of diversity.
- It is also called as “impurity reduction”.
- The attribute that maximizes the reduction in impurity (or equivalently, has the minimum value of) is selected for the attribute to be splitted.

n-ary Attribute Values to Binary Splitting

- The CART algorithm considers a binary split for each attribute.

- We shall discuss how the same is possible for attribute with more than two values.

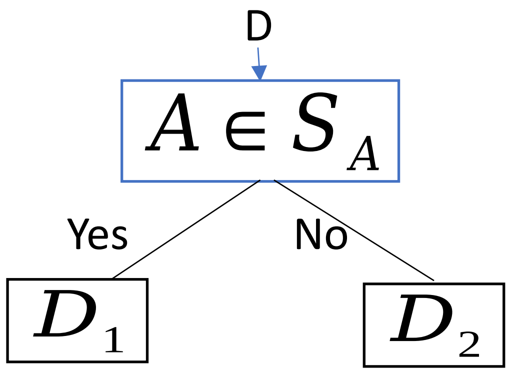
- **Case 1: Discrete valued attributes**
- Let us consider the case where A is a discrete-valued attribute having m discrete values .

- To determine the best binary split on A, we examine all of the possible subsets say of A that can be formed using the values of A.

- Each subset can be considered as a binary test for attribute A of the form .

n-ary Attribute Values to Binary Splitting

- Thus, given a data set *D*, we have to perform a test for an attribute value *A* like



- This test is satisfied if the value of *A* for the tuples is among the values listed in .

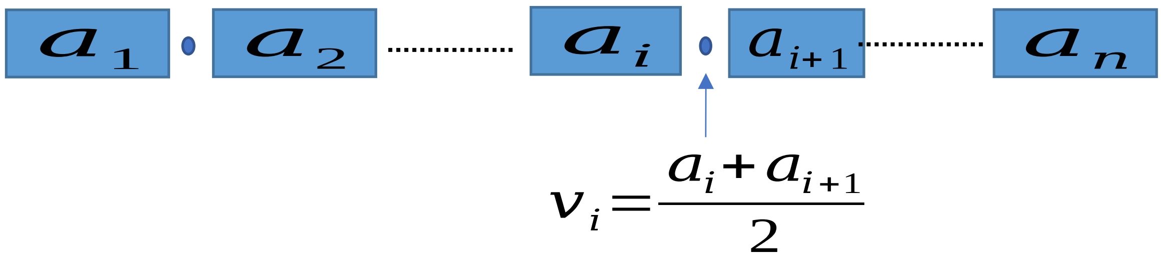
- If *A* has *m* distinct values in *D*, then there are possible subsets, out of which the empty subset and the power set should be excluded (as they really do not represent a split).

- Thus, there are possible ways to form two partitions of the dataset *D*, based on the binary split of *A*.

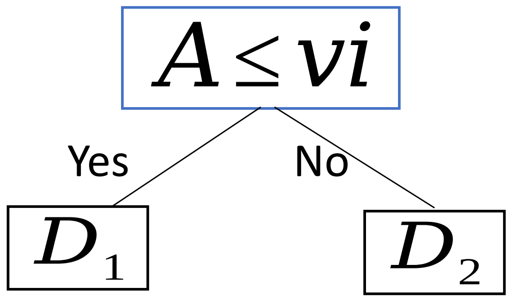
n-ary Attribute Values to Binary Splitting

Case2: Continuous valued attributes

- For a continuous-valued attribute, each possible split point must be taken into account.
- According to that strategy, the mid-point between a_i and a_{i+1} , let it be v_i, then



$$v_i = \frac{a_i + a_{i+1}}{2}$$



n-ary Attribute Values to Binary Splitting

- Each pair of (sorted) adjacent values is taken as a possible split-point say.
- is the set of tuples in *D* satisfying and in the set of tuples in D satisfying .
- The point giving the minimum Gini Index is taken as the split-point of the attribute *A*.

Note

- The attribute *A* and either its splitting subset (for discrete-valued splitting attribute) or split-point (for continuous valued splitting attribute) together form the splitting criteria.

CART Algorithm : Illustration

Example 20.1 : CART Algorithm

Suppose we want to build decision tree for the data set EMP as given in the table below.

Age
Y : young
M : middle-aged
O : old

Salary
L : low
M : medium
H : high

Job
G : government
P : private

Performance
A : Average
E : Excellent

Class : Select
Y : yes
N : no

Tuple#	Age	Salary	Job	Performance	Select
1	Y	H	P	A	N
2	Y	H	P	E	N
3	M	H	P	A	Y
4	O	M	P	A	Y
5	O	L	G	A	Y
6	O	L	G	E	N
7	M	L	G	E	Y
8	Y	M	P	A	N
9	Y	L	G	A	Y
10	O	M	G	A	Y
11	Y	M	G	E	Y
12	M	M	P	E	Y
13	M	H	G	A	Y
14	O	M	P	E	N

CART Algorithm : Illustration

For the EMP data set,

Now let us consider the calculation of for **Age**, **Salary**, **Job** and **Performance**.

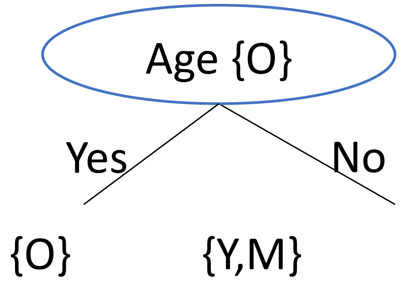
CART Algorithm : Illustration

Attribute of splitting: Age

The attribute age has three values, namely Y, M and O. So there are 6 subsets, that should be considered for splitting as:

?

?

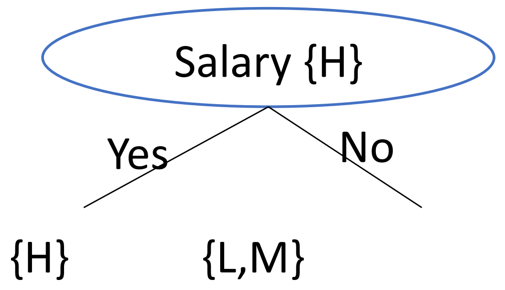


The best value of Gini Index while splitting attribute Age is

CART Algorithm : Illustration

Attribute of Splitting: Salary

The attribute salary has three values namely *L*, *M* and *H*. So, there are 6 subsets, that should be considered for splitting as:



CART Algorithm : Illustration

Attribute of Splitting: job

Job being the binary attribute , we have

?

?

CART Algorithm : Illustration

Attribute of Splitting: Performance

Job being the binary attribute , we have

?

?

Out of these gives the minimum value and hence, the attribute **Salary** would be chosen for splitting subset or .

Note:

It can be noted that the procedure following “information gain” calculation (i.e.) and that of “impurity reduction” calculation (i.e.) are near about.

Calculating γ using Frequency Table

- We have learnt that splitting on an attribute gives a reduction in the average Gini Index of the resulting subsets (as it does for entropy).

- Thus, in the same way the average weighted Gini Index can be calculated using the same frequency table used to calculate information gain which is as follows.

The G() for the subset

Calculating γ using Frequency Table

The average weighted Gini Index, () (assuming that attribute has *m* distinct values is)

The above gives a formula for *m*-attribute values; however, it an be fine tuned to subset of attributes also.

Illustration: Calculating γ using Frequency Table

Example 20.2 : Calculating γ using frequency table of OPTH

Let us consider the frequency table for OPTH database considered earlier. Also consider the attribute with three values 1, 2 and 3. The frequency table is shown below.

	1	**2**	**3**	**Row sum**
Class 1	2	1	1	4
Class 2	2	2	1	5
Class 3	4	5	6	15
Column sum	8	8	8	24

Illustration: Calculating γ using Frequency Table

Now we can calculate the value of Gini Index with the following steps:

1. For each non-empty column, form the sum of the squares of the values in the body of the table and divide by the column sum.

2. Add the values obtained for all columns and divided by , the size of the database.

3. Subtract the total from 1.

As an example, with reference to the frequency table as mentioned just.

75

So,

Illustration: Calculating γ using Frequency Table

Thus, the reduction in the value of Gini Index on splitting attribute is

where

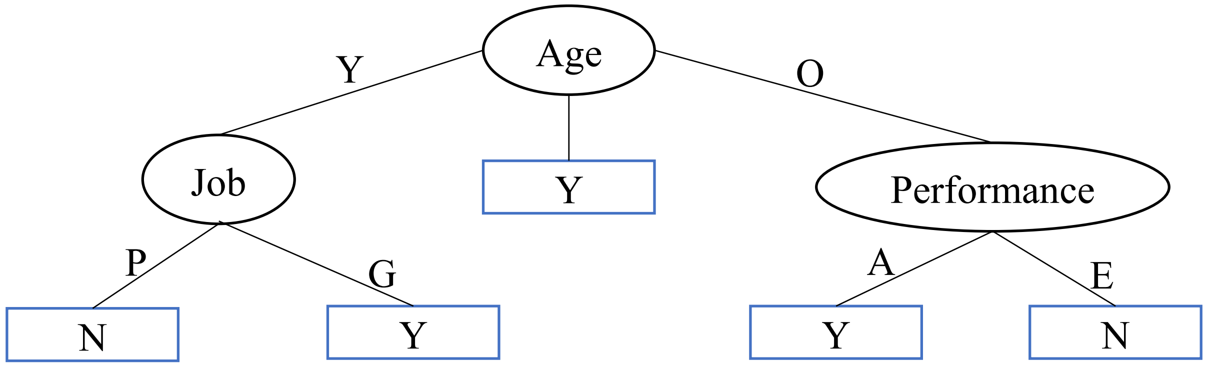
The calculation can be extended to other attributes in the OTPH database and is left as an exercise.

?

Decision Trees with ID3 and CART Algorithms

Example 20.3 : Comparing Decision Trees of EMP Data set

Compare two decision trees obtained using ID3 and CART for the EMP dataset. The decision tree according to ID3 is given for your ready reference (subject to the verification)



Decision Tree using ID3

?

Decision Tree using CART

Algorithm C4.5

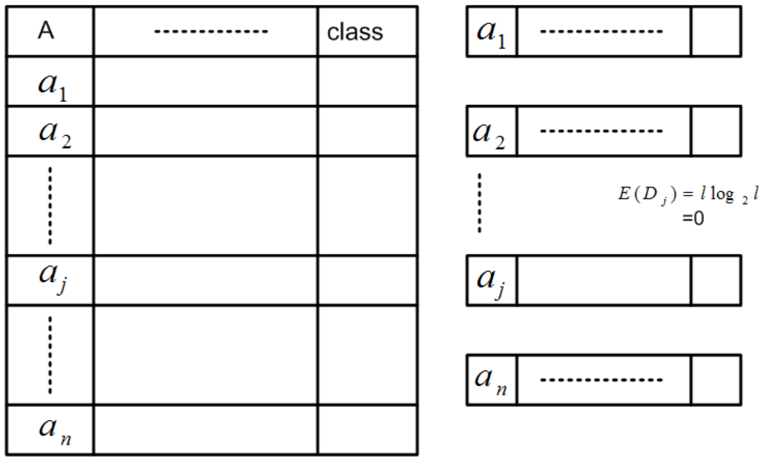
Algorithm C4.5 : Introduction

- J. Ross Quinlan, a researcher in machine learning developed a decision tree induction algorithm in 1984 known as ID3 (Iterative Dichotometer 3).
- Quinlan later presented C4.5, a successor of ID3, addressing some limitations in ID3.
- ID3 uses information gain measure, which is, in fact biased towards splitting attribute having a large number of outcomes.
- For example, if an attribute has distinct values for all tuples, then it would result in a large number of partitions, each one containing just one tuple.
    - In such a case, note that each partition is pure, and hence the purity measure of the partition, that is

Algorithm C4.5 : Introduction

Example 20.4 : Limitation of ID3

In the following, each tuple belongs to a unique class. The splitting on A is shown.



Thus, is maximum in such a situation.

Algorithm: C 4.5 : Introduction

- Although, the previous situation is an extreme case, intuitively, we can infer that ID3 favours splitting attributes having a large number of values
  - compared to other attributes, which have a less variations in their values.

- Such a partition appears to be useless for classification.

- This type of problem is called **overfitting problem**.

Note:

Decision Tree Induction Algorithm ID3 may suffer from overfitting problem.

Algorithm: C 4.5 : Introduction

- The overfitting problem in ID3 is due to the measurement of information gain.
- In order to reduce the effect of the use of the bias due to the use of information gain, C4.5 uses a different measure called **Gain Ratio**, denoted as .
- Gain Ratio is a kind of normalization to information gain using a **split information**.

Algorithm: C4.5 : Gain Ratio

Definition 20.3: **Gain Ratio**

The gain ratio can be defined as follows. We first define split information as

Here, m is the number of distinct values in A.

The gain ratio is then defined as where denotes the information gain on splitting the attribute A in the dataset D.

Physical Interpretation of (D)

(D)

- The value of split information depends on
  - the number of (distinct) values an attribute has and
  - how uniformly those values are distributed.
- In other words, it represents the potential information generated by splitting a data set D into m partitions, corresponding to the m outcomes of on attribute A.
- Note that for each outcome, it considers the number of tuples having that outcome with respect to the total number of tuples in D.

Physical Interpretation of (D)

Example 20.5 : Split information (D)

- To illustrate (D), let us examine the case where there are 32 instances and splitting an attribute *A* which has , , and sets of distinct values.
  - Distribution 1 : Highly non-uniform distribution of attribute values

Frequency	32	0	0	0

(D) = () 1 = 0

  - Distribution 2

Frequency	16	16	0	0

(D) = ()() = 1

Physical Interpretation of (D)

Distribution 3

Frequency	16	8	8	0

(D) = ()() 1.5

- Distribution 4

Frequency	16	8	4	4

- Distribution 5: Uniform distribution of attribute values

(D) = 1.75

Frequency	8	8	8	8

(D) = (())*4 = ().0

Physical Interpretation of (D)

- In general, if there are *m* attribute values, each occurring equally frequently, then the split information is .

- Based on the Example 20.5, we can summarize our observation on split information as under:

  - Split information is 0 when there is a single attribute value. It is a trivial case and implies *the minimum possible value of split information*.

  - For a given data set, when instances are uniformly distributed with respect to the attribute values, split information increases as the number of different attribute values increases.

  - The maximum value of split information occur when there are many possible attribute values, all are equally frequent.

Note:

- Split information varies between 0 and $log_2 m$ (both inclusive)

Physical Interpretation of

- Information gain signifies how much information will be gained on partitioning the values of attribute *A*
  - Higher information gain means splitting of *A* is more desirable.
- On the other hand, split information forms the denominator in the gain ratio formula.
  - This implies that higher the value of split information is, lower the gain ratio.
  - In turns, it decreases the information gain.
- Further, information gain is large when there are many distinct attribute values.
  - When many distinct values, split information is also a large value.
  - This way split information reduces the value of gain ratio, thus resulting a balanced value for information gain.
- Like information gain (in ID3), the attribute with the maximum gain ratio is selected as the splitting attribute in C4.5.

Calculation of using Frequency Table

- The frequency table can be used to calculate the gain ratio for a given data set and an attribute.
- We have already learned the calculation of information gain using Frequency Table.
- To calculate gain ratio, in addition to information gain, we are also to calculate split information.
- This split information can be calculated from frequency table as follows.
- For each non-zero column sum say contribute || for the *j*-th column (i.e., the *j*-th value of the attribute). Thus the split information is

(D) =

If there are *m*-columns in the frequency table.

Practice:

Using Gain ratio as the measurement of splitting attributes, draw the decision trees for OPTH and EMP data sets. Give calculation of gain ratio at each node.

Summary of Decision Tree Induction Algorithms

- We have learned the building of a decision tree given a training data.
  - The decision tree is then used to classify a test data.

- For a given training data *D*, the important task is to build the decision tree so that:
  - All test data can be classified accurately

  - The tree is balanced and with as minimum depth as possible, thus the classification can be done at a faster rate.

- In order to build a decision tree, several algorithms have been proposed. These algorithms differ from the chosen splitting criteria, so that they satisfy the above mentioned objectives as well as the decision tree can be induced with minimum time complexity. We have studied three decision tree induction algorithms namely ID3, CART and C4.5. A summary of these three algorithms is presented in the following table.

Table 20.6

Algorithm	Splitting Criteria	Remark
ID3	**Information Gain** (D) Where = Entropy of *D* (a measure of uncertainty) = where *D* is with set of *k* classes , , … , and = ; Here, is the set of tuples with class in *D.* (D) = Weighted average entropy when *D* is partitioned on the values of attribute A =) Here, *m* denotes the distinct values of attribute *A*.	• The algorithm calculates ,*D*) for all *in D* and choose that attribute which has **maximum** ,*D*). • The algorithm can handle both categorical and numerical attributes. • It favors splitting those attributes, which has a large number of distinct values.

Algorithm	Splitting Criteria	Remark
CART	**Gini Index** (D) where = Gini index (a measure of impurity) = Here, = and D is with k number of classes $G_A(D) =$) +), when D is partitioned into two data sets and based on some values of attribute A.	• The algorithm calculates all binary partitions for all possible values of attribute A and choose that binary partition which has the **maximum** • The algorithm is computationally very expensive when the attribute A has a large number of values.

Algorithm	Splitting Criteria	Remark
C4.5	**Gain Ratio** where = Information gain of *D* (same as in ID3, and = splitting information = when *D* is partitioned into , , … , partitions corresponding to *m* distinct attribute values of *A*.	• The attribute *A* with **maximum** value of is selected for splitting. • Splitting information is a kind of normalization, so that it can check the biasness of information gain towards the choosing attributes with a large number of distinct values.

In addition to this, we also highlight few important characteristics of decision tree induction algorithms in the following.

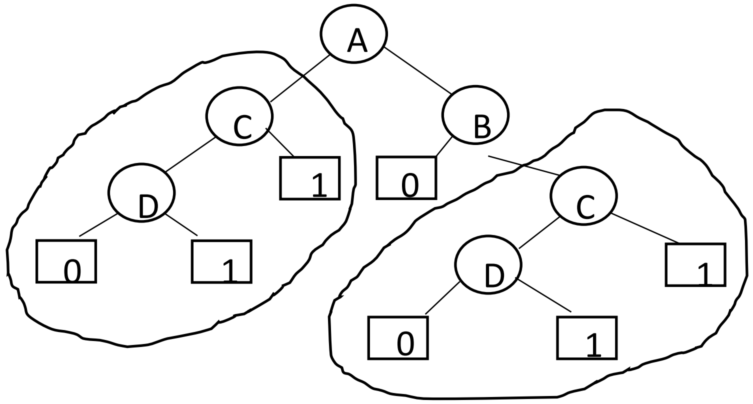
Notes on Decision Tree Induction algorithms

1. **Optimal Decision Tree:** Finding an optimal decision tree is an NP-complete problem. Hence, decision tree induction algorithms employ a heuristic based approach to search for the best in a large search space. Majority of the algorithms follow a greedy, top-down recursive divide-and-conquer strategy to build decision trees.
2. **Missing data and noise:** Decision tree induction algorithms are quite robust to the data set with missing values and presence of noise. However, proper data pre-processing can be followed to nullify these discrepancies.
3. **Redundant Attributes:** The presence of redundant attributes does not adversely affect the accuracy of decision trees. It is observed that if an attribute is chosen for splitting, then another attribute which is redundant is unlikely to chosen for splitting.
4. **Computational complexity:** Decision tree induction algorithms are computationally inexpensive, in particular, when the sizes of training sets are large, Moreover, once a decision tree is known, classifying a test record is extremely fast, with a worst-case time complexity of O(d), where d is the maximum depth of the tree.

Notes on Decision Tree Induction algorithms

5. **Data Fragmentation Problem:** Since the decision tree induction algorithms employ a top-down, recursive partitioning approach, the number of tuples becomes smaller as we traverse down the tree. At a time, the number of tuples may be too small to make a decision about the class representation, such a problem is known as the data fragmentation. To deal with this problem, further splitting can be stopped when the number of records falls below a certain threshold.

6. **Tree Pruning:** A sub-tree can replicate two or more times in a decision tree (see figure below). This makes a decision tree unambiguous to classify a test record. To avoid such a sub-tree replication problem, all sub-trees except one can be pruned from the tree.



Reference

- The detail material related to this lecture can be found in

  Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2015.

  Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, Addison-Wesley, 2014

Any question?

Feature Translation

More Regression and Classification approaches

Linear Regression

- Given data with n dimensional variables and 1 target-variable (real number)

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$$

Where $\mathbf{x} \in \Re^n, y \in \Re$

- The objective: Find a function f that returns the best fit. $f : \Re^n \to \Re$
- Assume that the relationship between X and y is approximately linear. The model can be represented as (w represents coefficients and b is an intercept)

$$f(w_1, \ldots, w_n, b) = y = \mathbf{w} \cdot \mathbf{x} + b + \varepsilon$$

Linear Regression

- To find the best fit, we minimize the sum of squared errors → Least square estimation

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - (\mathbf{w} \cdot \mathbf{x}_i + b))^2$$

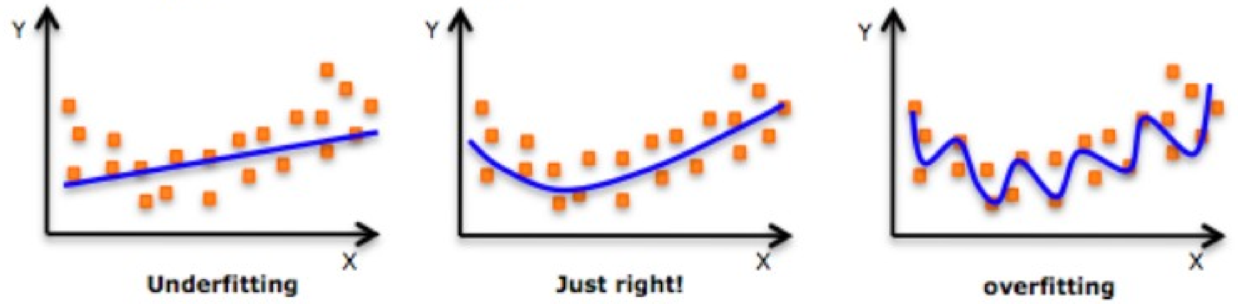
- The solution can be found by solving

  $\hat{\mathbf{w}} = (X^T X)^{-1} X^T Y$

  (By taking the derivative of the above objective function w.r.t. $\mathbf{w}$)
- In MATLAB, the back-slash operator computes a least square solution.

Linear Regression

$$\min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} (y_i - (\hat{\mathbf{w}} \cdot \mathbf{x}_i + \hat{b}))^2$$



- To ovoid over-fitting, a regularization term can be introduced (minimize a magnitude of w)
  - LASSO: $$\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} |w_j|$$
  - Ridge regression: $$\min \sum_{i=1}^{m} (y_i - \mathbf{w} \cdot \mathbf{x}_i - b)^2 + C \sum_{j=1}^{n} |\mathbf{w}_j^2|$$

Support Vector Regression

- Find a function, f(x), with at most ε-deviation from the target y

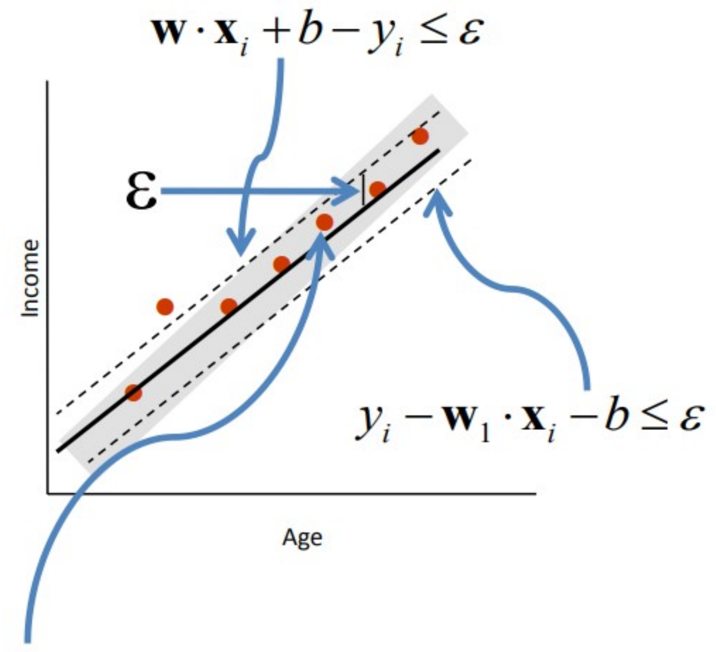
The problem can be written as a convex optimization problem

$$\min \frac{1}{2} \| \mathbf{w} \|^2$$
$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$
$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

C: trade off the complexity

What if the problem is not feasible?

We can introduce slack variables (similar to soft margin loss function).



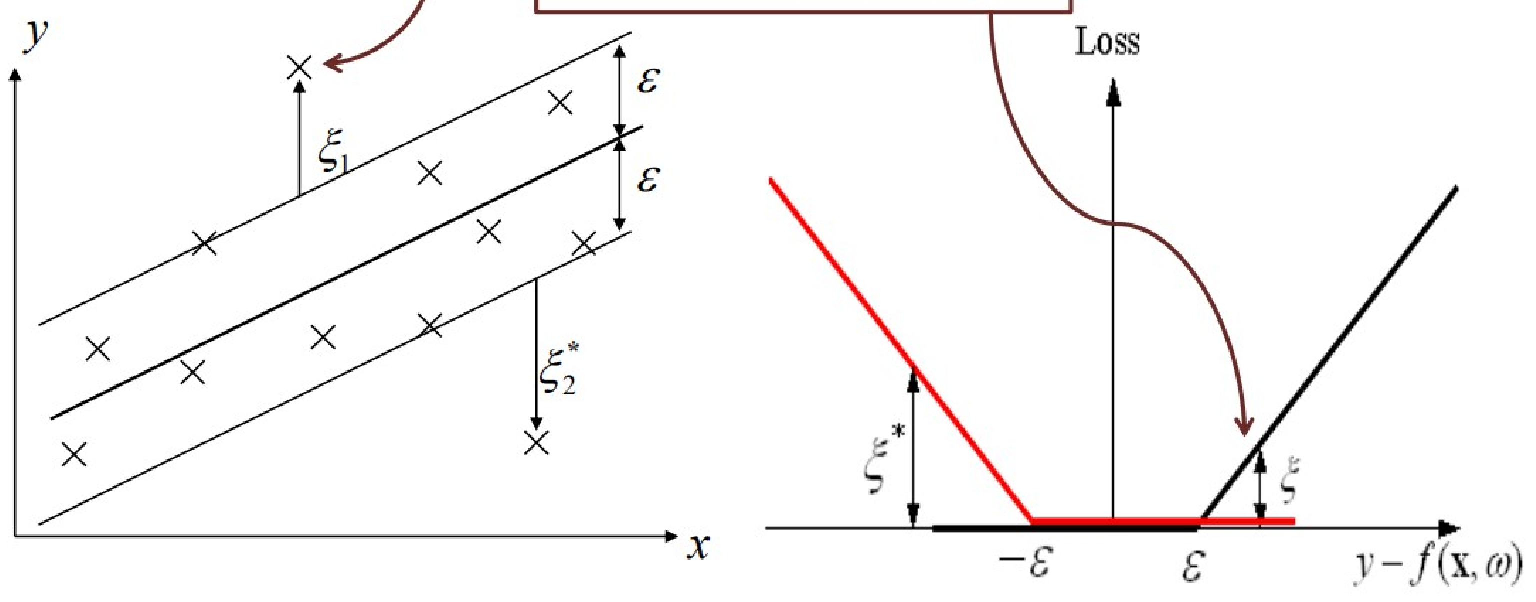
$\mathbf{w} \cdot \mathbf{x}_i + b - y_i \le \varepsilon$
ε
Income
Age
$y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon$

We do not care about errors as long as they are less than ε

Support Vector Regression

Assume linear parameterization $f(\mathbf{x},\omega) = \mathbf{w}\cdot\mathbf{x} + b$

Only the point outside the ε-region contribute to the final cost

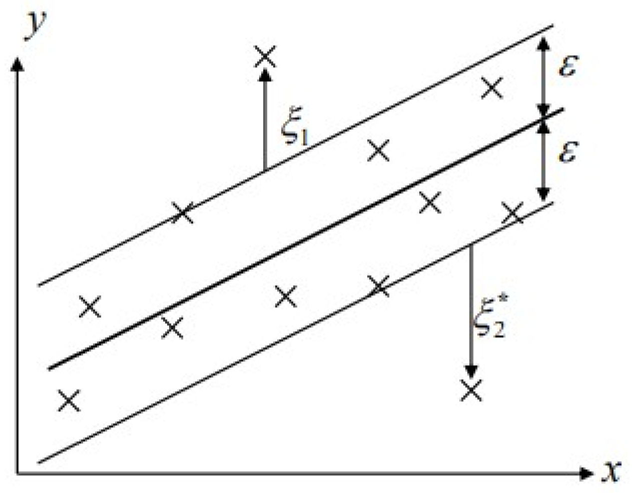


$$L_\varepsilon(y, f(\mathbf{x},\omega)) = \max\left(|\, y - f(\mathbf{x},\omega)\,| - \varepsilon, 0\right)$$

Soft margin

Given training data

$$\left(\mathbf{x}_i, y_i\right) \quad i = 1, \ldots, m$$



Minimize

$$\frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*)$$

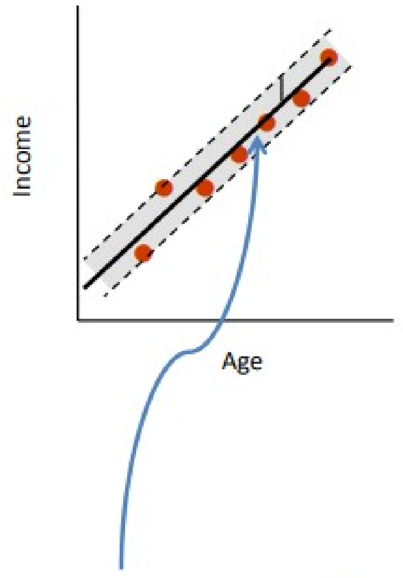
Under constraints

$$\begin{cases} y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b \le \varepsilon + \xi_i \\ (\mathbf{w} \cdot \mathbf{x}_i) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i \ , \xi_i^* \ge 0, i = 1, \ldots, m \end{cases}$$

Linear versus Non-linear SVR

- **Linear case**

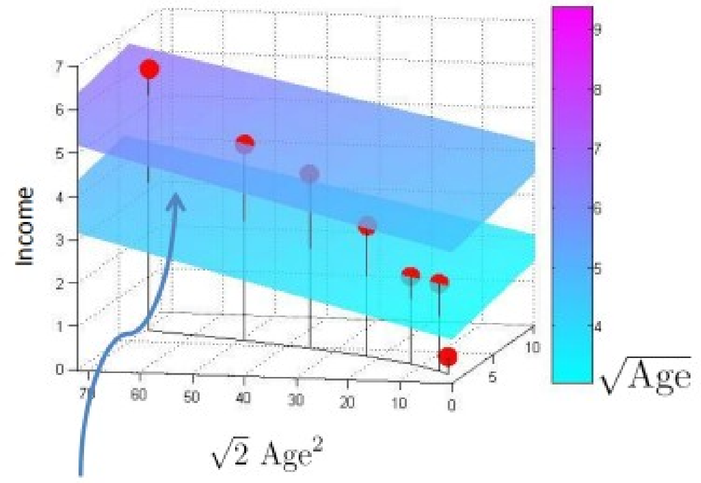
$$f : age \rightarrow income$$



$$y_i = \mathbf{w}_1 \cdot \mathbf{x}_i + b$$

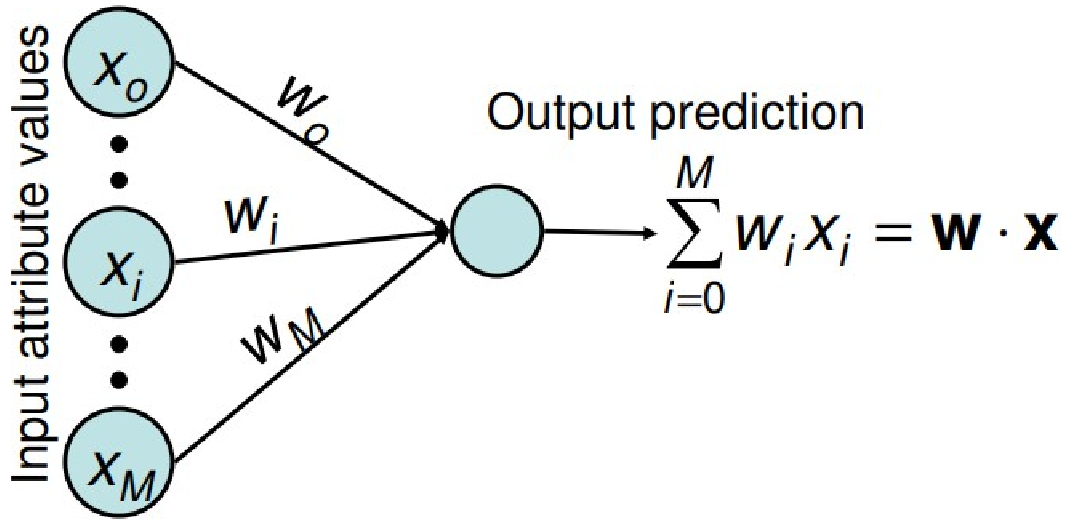
- **Non-linear case**
  - Map data into a higher dimensional space, e.g.,

$$f : (\sqrt{age}, \sqrt{2}age^2) \rightarrow income$$

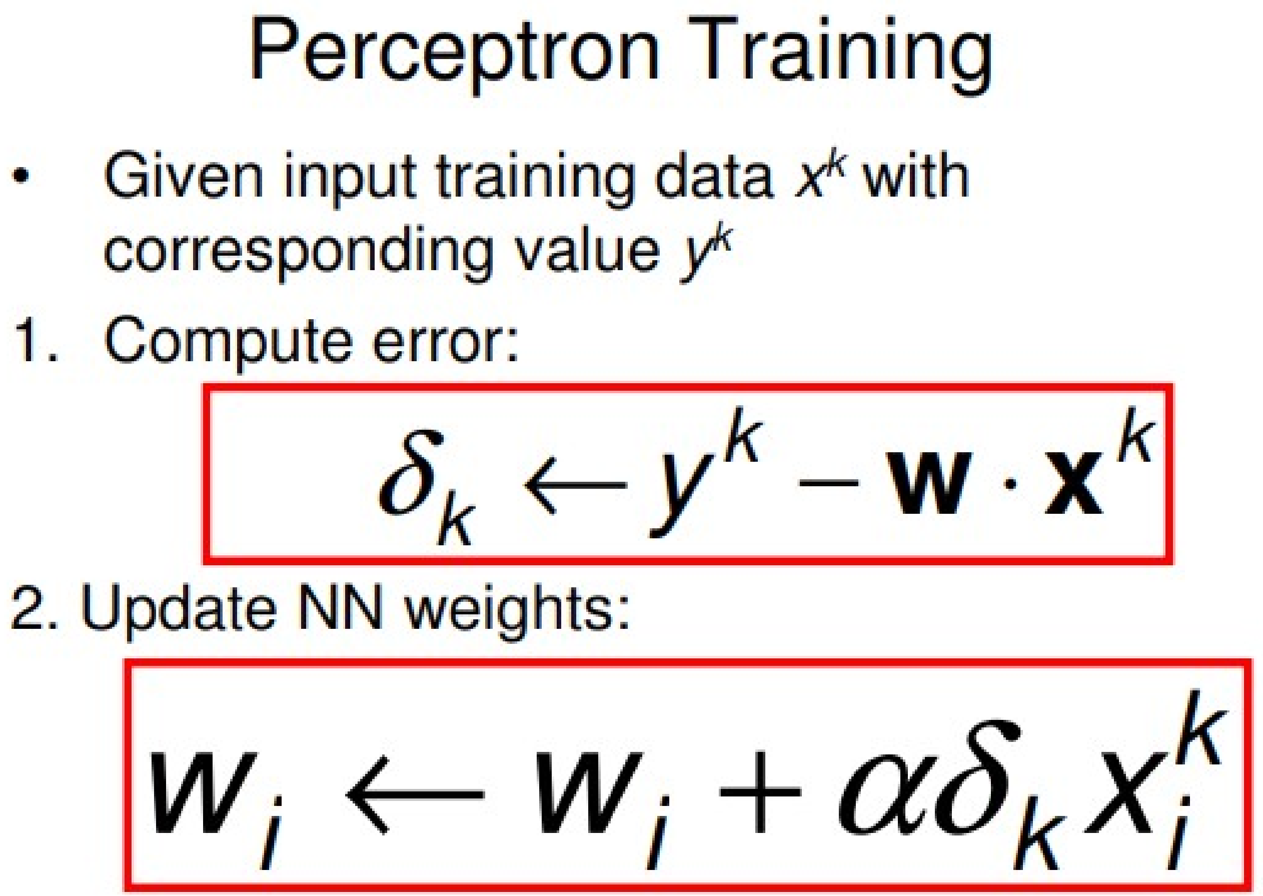


$$y_i = \mathbf{w}_1 \sqrt{\mathbf{x}_i} + \mathbf{w}_2 \sqrt{2}\mathbf{x}_i^2 + b$$

Regression: Neural Networks



Regression: Neural Networks



Perceptron Training

- Given input training data x^k with corresponding value y^k

1. Compute error:

$$\boxed{\delta_k \leftarrow y^k - \mathbf{w} \cdot \mathbf{x}^k}$$

2. Update NN weights:

$$\boxed{w_i \leftarrow w_i + \alpha \delta_k x_i^k}$$

Regression: Neural Networks

- Update has many names: delta rule, gradient rule, LMS rule
- Update is guaranteed to converge to the best linear fit (global minimum of E)
- Of course, there are more direct ways of solving the linear regression problem by using linear algebra techniques. It boils down to a simple matrix inversion (not shown here).
- In fact, the perceptron training algorithm can be much, much slower than the direct solution
- Use of MLP with Backpropagation for Non-Linear Regression

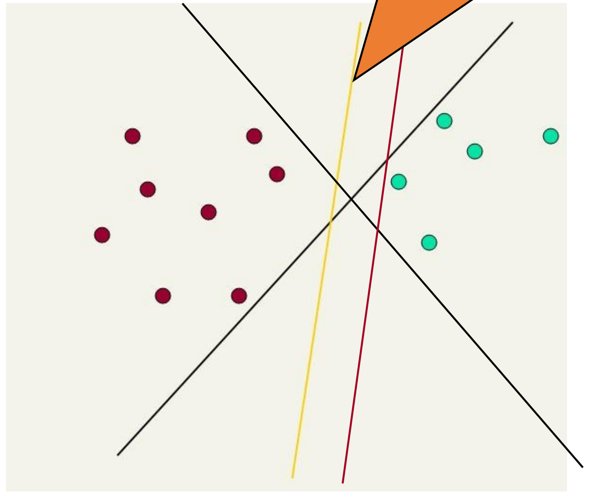
Perceptron Learning Algorithm

- First neural network learning model in the 1960’s
- Simple and limited (single layer model)
- Basic concepts are similar for multi-layer models, so this is a good learning tool
- Still used in some current applications (large business problems, where intelligibility is needed, etc.)

Linear classifiers: Which Hyperplane?

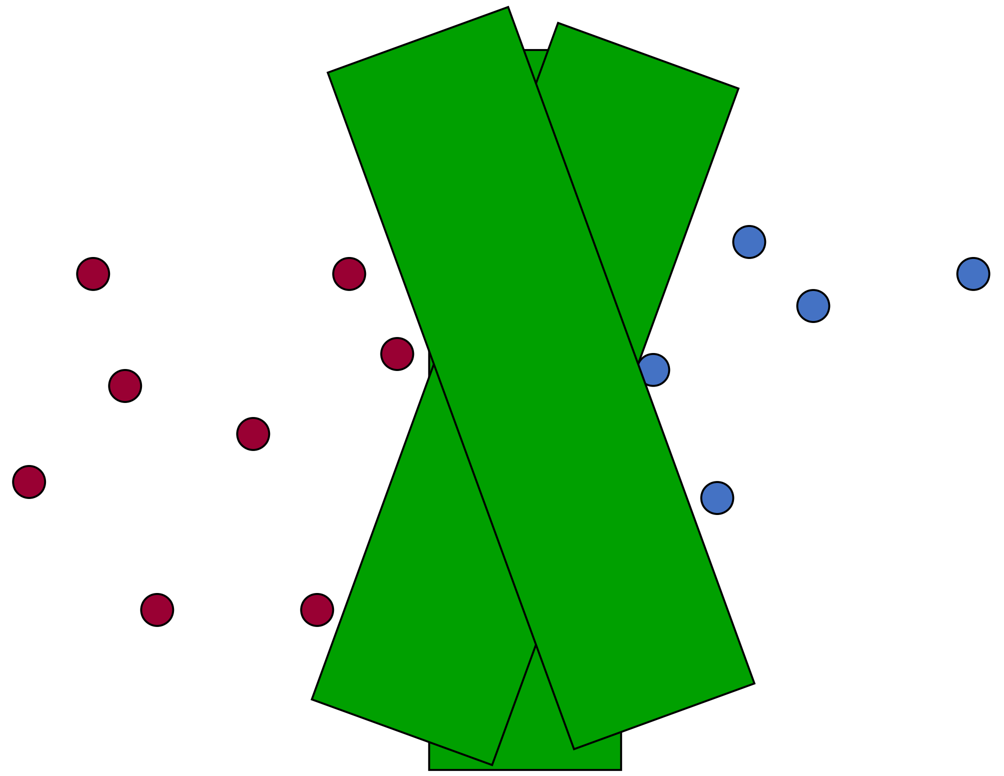
- Lots of possible solutions for *a, b, c.*
- Some methods find a separating hyperplane, but not the optimal one [according to some criterion of expected goodness]
  - E.g., perceptron
- Support Vector Machine (SVM) finds an optimal* solution.
  - Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary
  - One intuition: if there are no points near the decision surface, then there are no very uncertain classification decisions

This line represents the decision boundary:
$ax + by - c = 0$



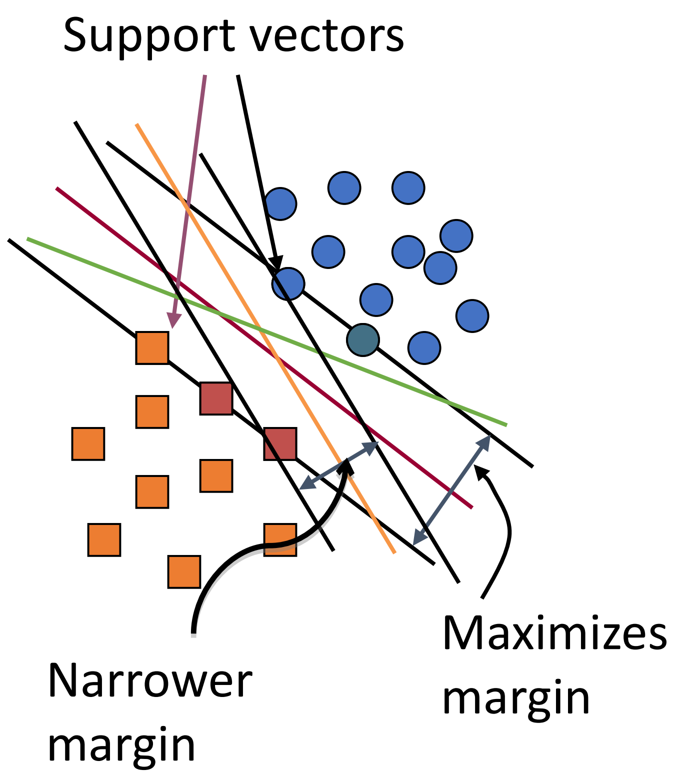
Another intuition

- If you have to place a fat separator between classes, you have less choices, and so the capacity of the model has been decreased



Support Vector Machine (SVM)

- SVMs maximize the *margin* around the separating hyperplane.
    - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- Solving SVMs is a *quadratic programming* problem
- Seen by many as the most successful current text classification method*



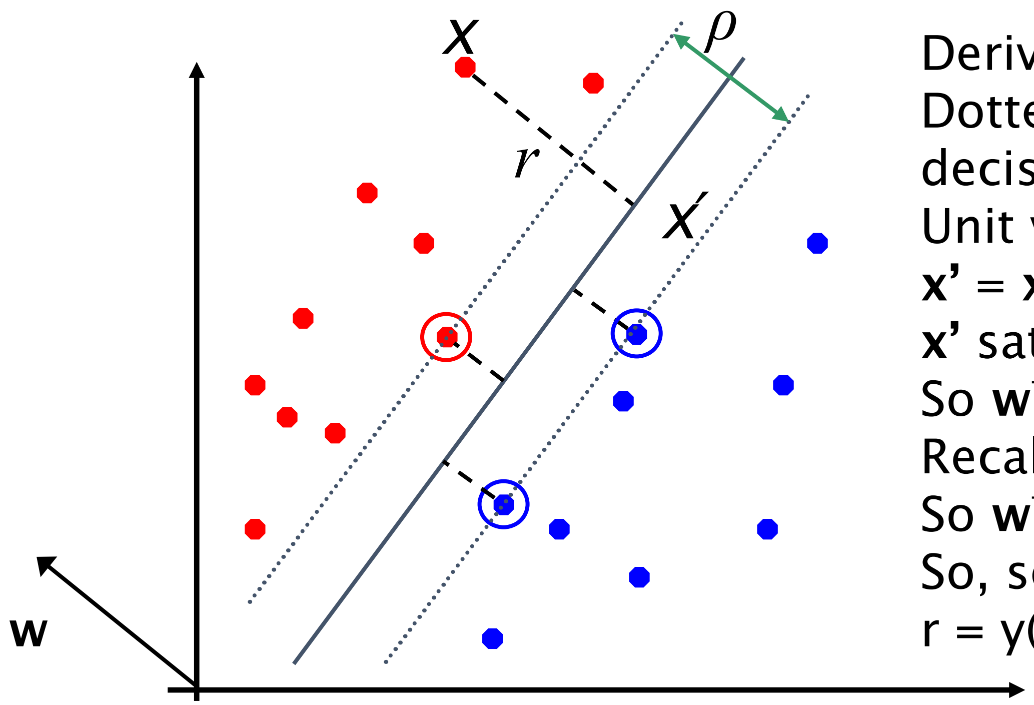
*but other discriminative methods often perform very similarly

Maximum Margin: Formalization

- $\mathbf{w}$: decision hyperplane normal vector
- $\mathbf{x}_i$: data point *i*
- y_i: class of data point *i* (+1 or -1) NB: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $y_i\,(\mathbf{w}^T\mathbf{x}_i + b)$
  - But note that we can increase this margin simply by scaling **w**, **b**….
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

Geometric Margin

- Distance from example to the separator is $$r = y\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$$
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.



Derivation of finding *r:*
Dotted line **x'**–**x** is perpendicular to decision boundary so parallel to **w**.
Unit vector is **w**/|**w**|, so line is r**w**/|**w**|.
x' = **x** – yr**w**/|**w**|.
x' satisfies $\mathbf{w}^T\mathbf{x}'$+b = 0.
So $\mathbf{w}^T$(**x** –yr**w**/|**w**|) + b = 0
Recall that |**w**| = sqrt($\mathbf{w}^T\mathbf{w}$).
So $\mathbf{w}^T\mathbf{x}$ –yr|**w**| + b = 0
So, solving for r gives:
r = y($\mathbf{w}^T\mathbf{x}$ + b)/|**w**|

Linear SVM Mathematically

The linearly separable case

- Assume that all data is at least distance 1 from the hyperplane, then the following two constraints follow for a training set $\{(\mathbf{x_i}, y_i)\}$

$$\mathbf{w^T x_i} + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w^T x_i} + b \leq -1 \quad \text{if } y_i = -1$$

- For support vectors, the inequality becomes an equality
- Then, since each example’s distance from the hyperplane is

$$r = y\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

- The margin is:

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Linear Support Vector Machine (SVM)

- **Hyperplane**

  $\mathbf{w}^T \mathbf{x} + b = 0$

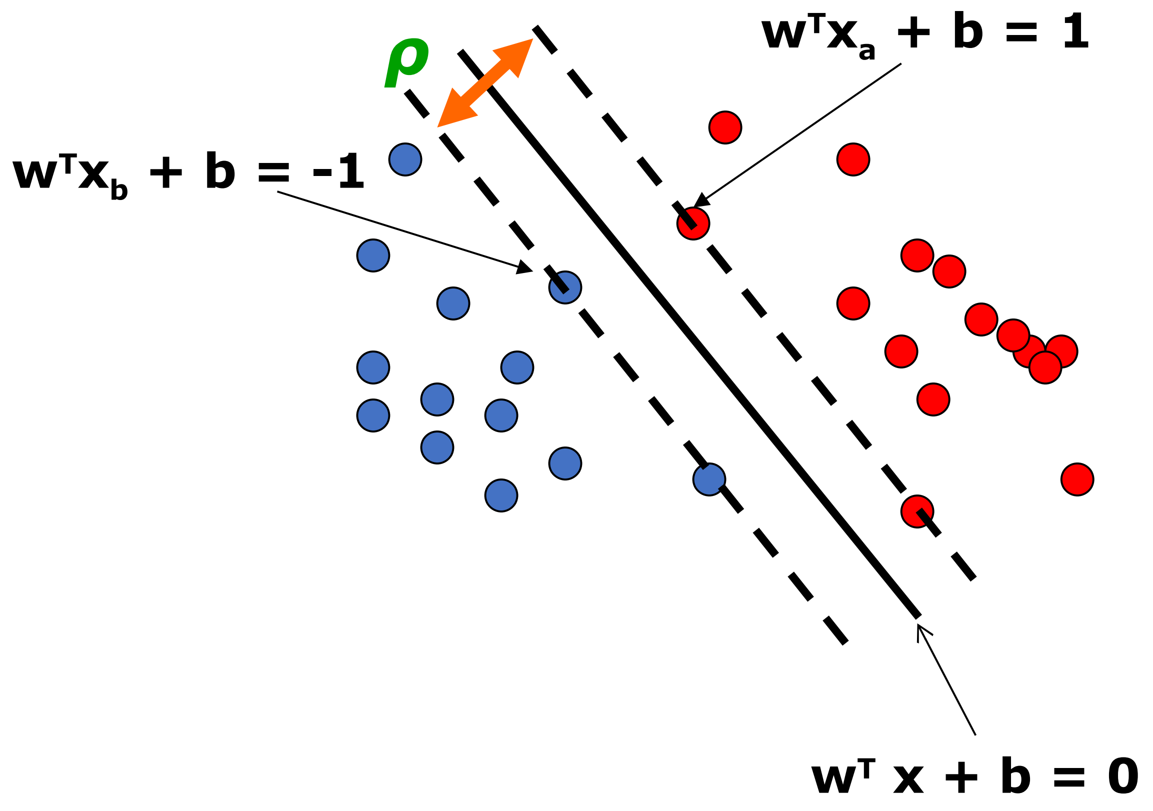
- **Extra scale constraint**:

  $\min_{i=1,\ldots,n} |\mathbf{w}^T\mathbf{x}_i + \mathbf{b}| = 1$

- This implies:

  $w^T(x_a - x_b) = 2$

  $\rho = ||x_a - x_b||_2 = 2/||\mathbf{w}||_2$



ρ
$\mathbf{w}^T\mathbf{x}_a + \mathbf{b} = 1$
$\mathbf{w}^T\mathbf{x}_b + \mathbf{b} = -1$
$\mathbf{w}^T \mathbf{x} + \mathbf{b} = 0$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

> Find $\mathbf{w}$ and b such that
>
> $\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$
>
> $\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min **||w||** = max 1/ **||w||**):

> Find $\mathbf{w}$ and b such that
>
> $\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;
>
> and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

- This is now optimizing a *quadratic* function subject to *linear* constraints
- Quadratic optimization problems are a well-known class of mathematical programming problem, and many (intricate) algorithms exist for solving them (with many special ones built for SVMs)
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^\mathbf{T}\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

- The solution has the form:

$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i} \qquad b = y_k - \mathbf{w^T x_k} \text{ for any } \mathbf{x_k} \text{ such that } \alpha_k \neq 0}$$

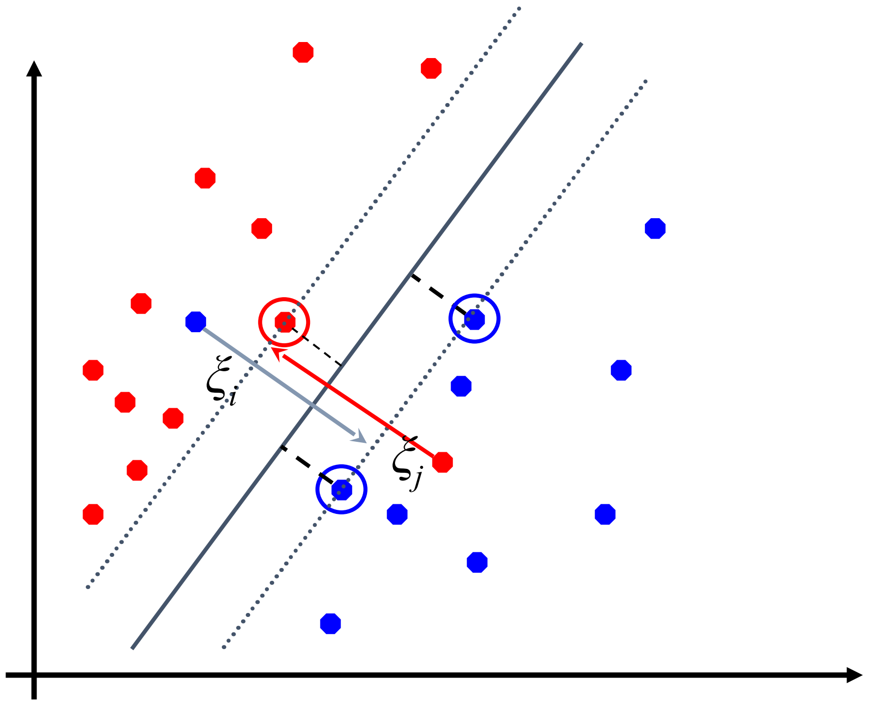
- Each non-zero α_i indicates that corresponding $\mathbf{x_i}$ is a support vector.
- Then the classifying function will have the form:

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i^T x} + b}$$

- Notice that it relies on an *inner product* between the test point **x** and the support vectors $\mathbf{x_i}$
  - We will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x_i^T x_j}$ between all pairs of training points.

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find **w** and b such that
$\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + \mathrm{b}) \geq 1$

- The new formulation incorporating slack variables:

Find **w** and b such that
$\boldsymbol{\Phi}(\mathbf{w}) = \frac{1}{2}\,\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^{\mathbf{T}}\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter *C* can be viewed as a way to control overfitting
    - A regularization term

Soft Margin Classification – Solution

- The dual problem for soft margin classification:

Find $\alpha_1 \ldots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $0 \leq \alpha_i \leq C$ for all α_i

- Neither slack variables ξ_i nor their Lagrange multipliers appear in the dual problem!
- Again, $\mathbf{x_i}$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x_i}$

$b = y_k(1 - \xi_k) - \mathbf{w}^T\mathbf{x}_k$ where $k = \underset{k'}{\text{argmax}}\ \alpha_{k'}$

w is not needed explicitly for classification!

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x_i}^T\mathbf{x} + b$

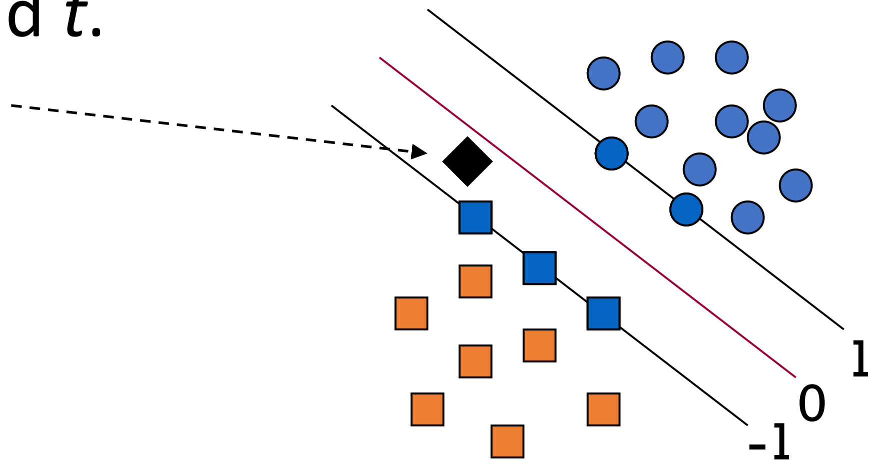
Classification with SVMs

- Given a new point **x**, we can score its projection onto the hyperplane normal:
  - I.e., compute score: $\mathbf{w}^T\mathbf{x} + b = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$
    - Decide class based on whether < or > 0
  - Can set confidence threshold *t*.

Score > *t*: yes

Score < -*t*: no

Else: don't know



Linear SVMs: Summary

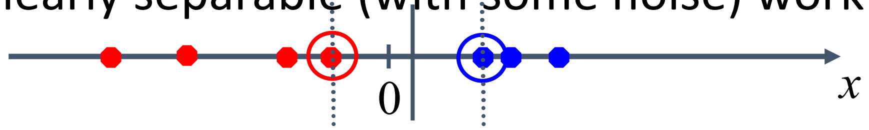
- The classifier is a *separating hyperplane.*
- The most “important” training points are the support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $\mathbf{x}_i$ are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution, training points appear only inside inner products:

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x}_i^T\mathbf{x}_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

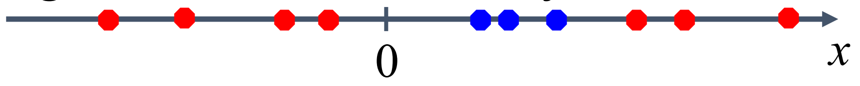
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x}_i^T\mathbf{x}} + b$$

Non-linear SVMs

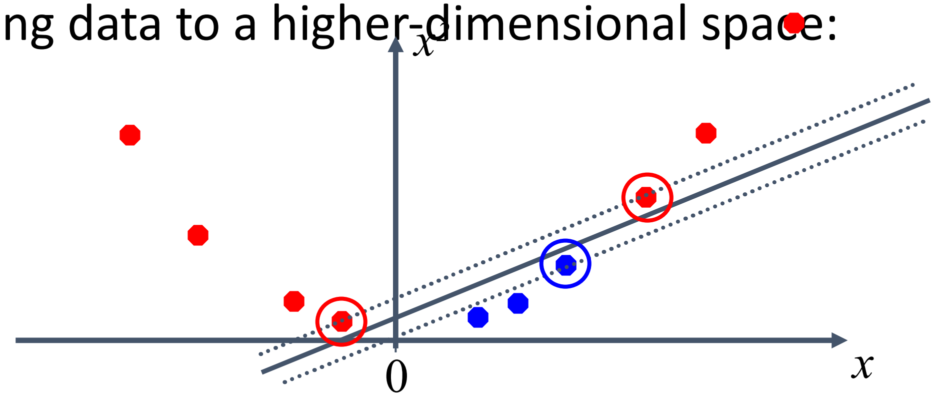
- Datasets that are linearly separable (with some noise) work out great:



- But what are we going to do if the dataset is just too hard?

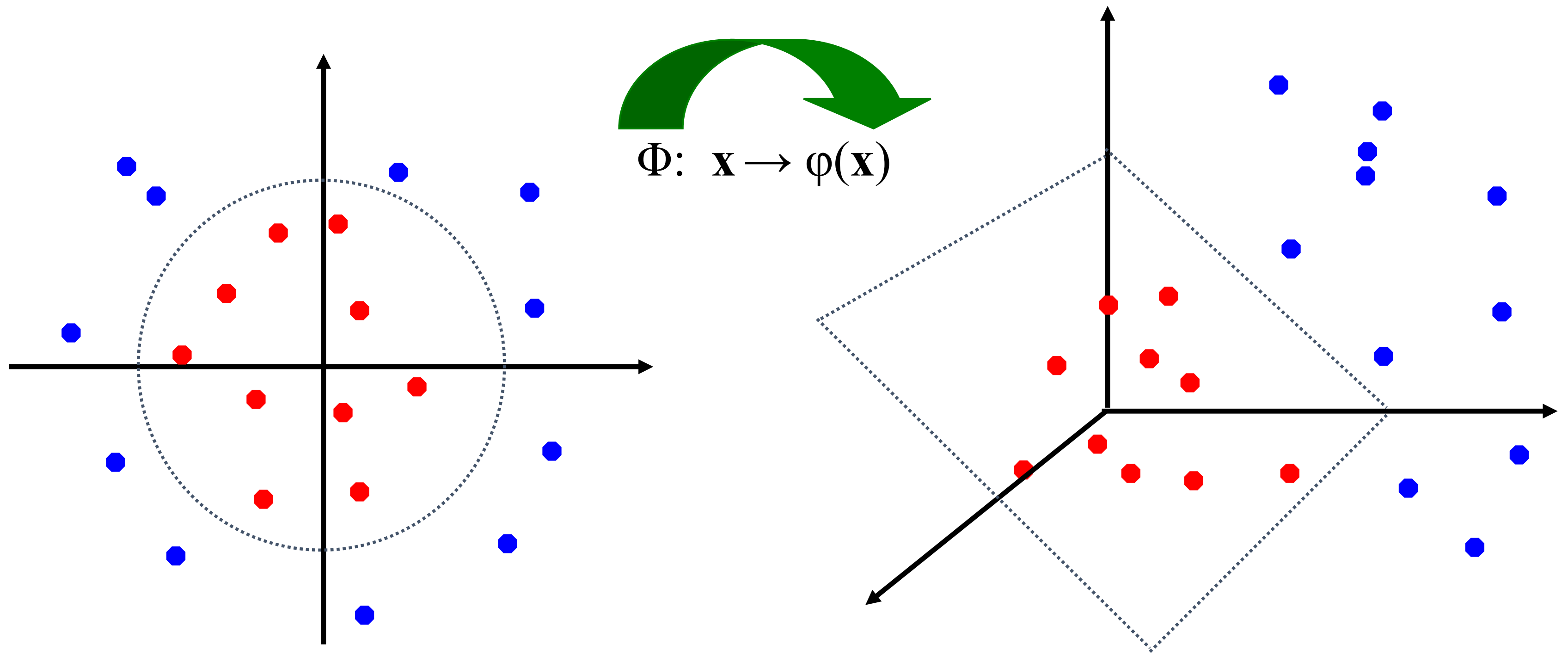


- How about … mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on an inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1\ \ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^T\mathbf{x}_j)^2 ,= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1\ \ x_{i1}^2\ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T\, [1\ \ x_{j1}^2\ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]$

  $= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ where $\phi(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \sqrt{2}x_2]$

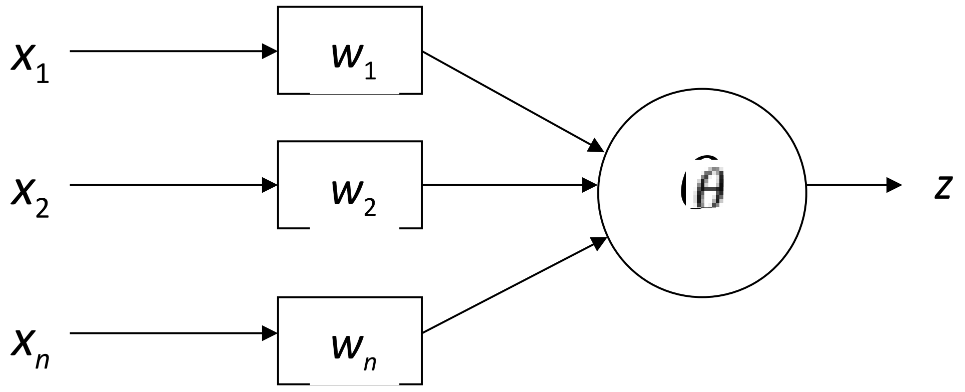
Kernels

- Why use kernels?
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear
  - Polynomial **$K(x,z) = (1+x^Tz)^d$**
    - Gives feature conjunctions
  - Radial basis function (infinite dimensional space)

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2}$$

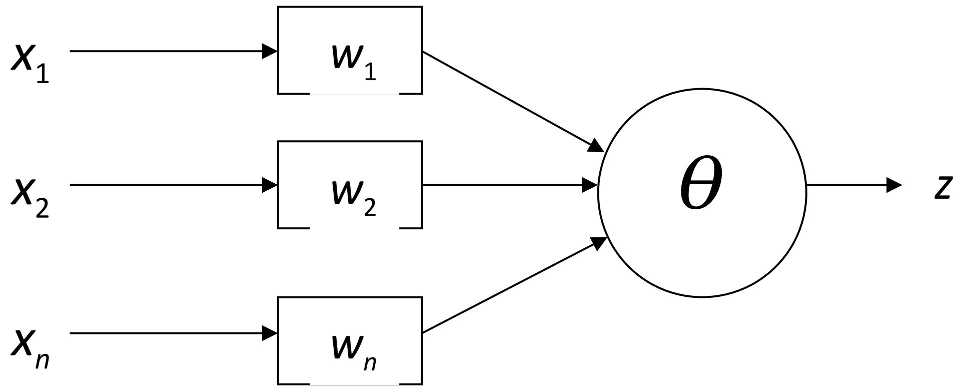
- Haven't been very useful in text classification

Classification: Perceptron Node – Threshold Logic Unit



$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$

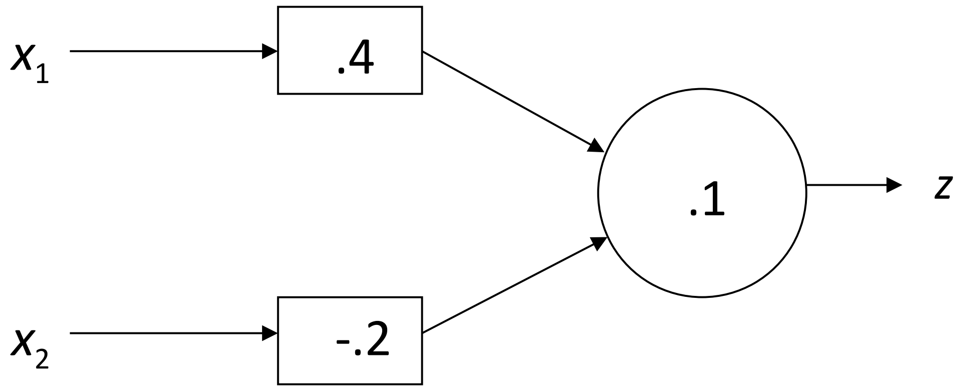
Perceptron Node – Threshold Logic Unit



- Learn weights such that an objective function is maximized.
- What objective function should we use?
- What learning algorithm should we use?

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$

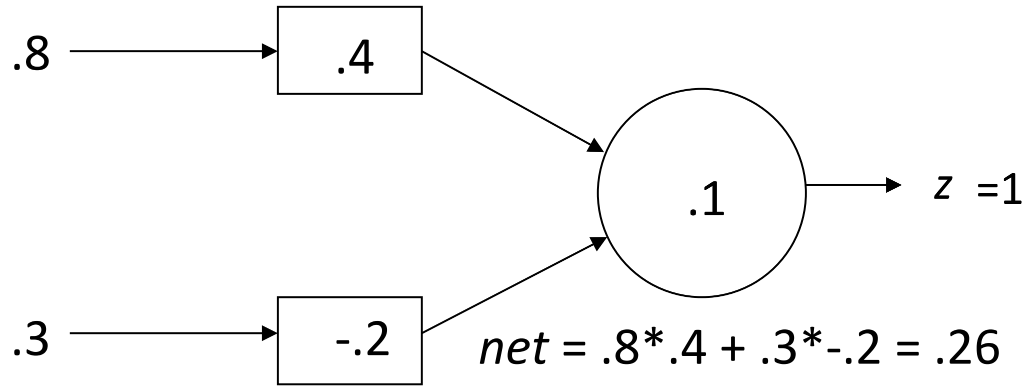
Perceptron Learning Algorithm



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$

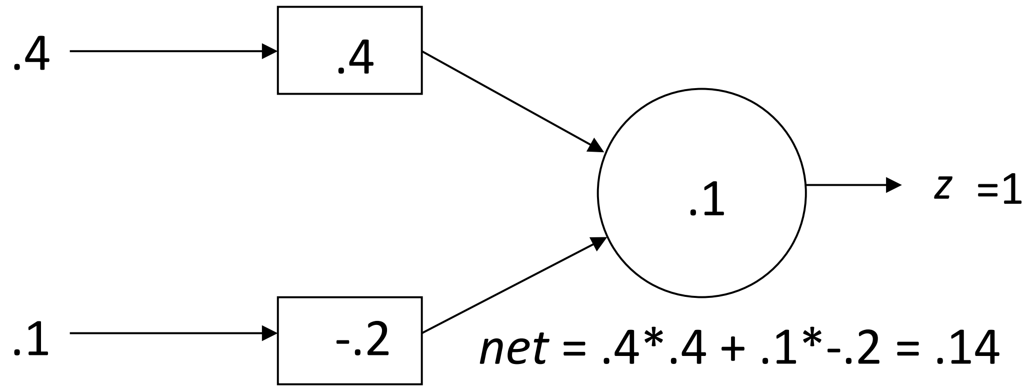
First Training Instance



x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$

Second Training Instance



net = .4*.4 + .1*-.2 = .14

x_1	x_2	t
.8	.3	1
.4	.1	0

$$z = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} x_i w_i < \theta \end{cases}$$

$$\Delta w_i = (t - z) * c * x_i$$

Perceptron Rule Learning

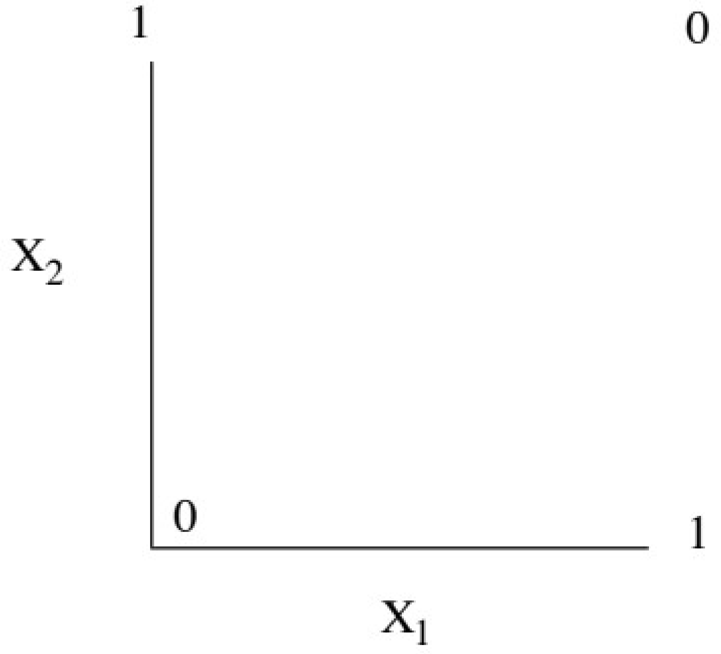
$$\Delta w_i = c(t - z)\, x_i$$

- Where w_i is the weight from input *i* to perceptron node, *c* is the learning rate, *t* is the target for the current instance, *z* is the current output, and x_i is ith input
- Least perturbation principle
  - Only change weights if there is an error
  - small *c* rather than changing weights sufficient to make current pattern correct
  - Scale by x_i
- Create a perceptron node with *n* inputs
- Iteratively apply a pattern from the training set and apply the perceptron rule
- Each iteration through the training set is an *epoch*
- Continue training until total training set error ceases to improve
- Perceptron Convergence Theorem: Guaranteed to find a solution in finite time if a solution exists

How to Handle Multi-Class Output

- This is an issue with any learning model which only supports binary classification (perceptron, SVM, etc.)
- Create 1 perceptron for each output class, where the training set considers all other classes to be negative examples **(one vs the rest)**
  - Run all perceptrons on novel data and set the output to the class of the perceptron which outputs high
  - If there is a tie, choose the perceptron with the highest net value
- Create 1 perceptron for each pair of output classes, where the training set only contains examples from the 2 classes **(one vs one)**
  - Run all perceptrons on novel data and set the output to be the class with the most wins (votes) from the perceptrons
  - In case of a tie, use the net values to decide
  - Number of models grows by the square of the output classes

Exclusive Or



Is there a dividing hyperplane?